

The New Pinscape Build Guide

by Michael Roberts

First edition, October 2019

Revised and updated from time to time, most recently in June, 2024

Copyright and license

Copyright ©2016-2024 Michael J. Roberts.

This book is licensed under a Creative Commons Attribution-ShareAlike 4.0

International License. In brief, this means

that you're free to use, share, and adapt this work for any purpose, without seeking further permission from the author, as long as you give appropriate credit and pass along the same permissions in your adapted material.



The Pinscape Controller firmware and the Expansion Board hardware designs have similar Open Source license terms. See their respective license files for details.

No Warranty

Just to be sure there's no misunderstanding, I want to spell out that this documentation and the related mechanical and electronic designs and computer software have NO WARRANTY of any kind.

This entire enterprise is a hobby project, and I don't have a Quality Assurance department to independently test any of it, so the material undoubtedly contains numerous errors. I'm making all of this available at no charge in the hope that it's useful, but I can't guarantee that it'll work or that you'll be successful building any of it. **Work involving electronic circuitry has the potential to damage other connected equipment, including expensive items such as computer motherboards. Proceed at your own risk.**



THIS WORK IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE WORK OR THE USE OR OTHER DEALINGS IN THE WORK.

Caution

Building a virtual pinball machine involves some inherently dangerous equipment and activities, such as power tools and high voltages. Read the manufacturer's instruction manuals for your tools and follow their safety precautions. Wear safety glasses, hearing protection, and other appropriate safety gear for each task. Don't attempt anything that you don't understand or don't feel comfortable with.

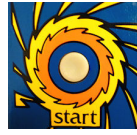


Exercise extreme caution when working with electricity, especially with high voltages. Higher voltages can cause electrocution, and even low voltages can start fires and damage other equipment. Always disconnect the power at its source before doing work on anything electrical. Don't just switch it off - unplug it from the wall outlet.

Contents

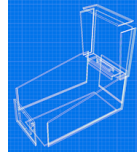
Part One. *Preliminaries*

1. Preface
2. What is a pin cab?
3. A Visual Guide to the Virtual Pinball Cabinet
4. Resources



Part Two. *Planning and Building the Cabinet*

5. Road Map
6. Serviceable Design
7. Selecting a Playfield TV
8. Selecting a Backbox TV
9. Selecting a DMD Device
10. Designing the PC
11. Power Switching
12. I/O Controllers
13. PC Hardware Setup
14. Windows Setup
15. Pinball Software Setup
16. Backglass Software Setup
17. Optimizing Performance
18. Customizing VP Tables
19. Cabinet Building Tools
20. Cabinet Parts List
21. Cabinet Body
22. Cabinet Art
23. Cabinet Hardware Installation
24. Backbox Hardware Installation
25. Cabinet Grounding
26. Inside the Cabinet
27. Installing the PC
28. Cooling Fans
29. Playfield TV Mounting
30. Backbox TV Mounting



31. Speaker/DMD Panel
32. Original WPC Speaker Panel
33. WPC-95 Speaker Panel
34. Cabinet Buttons
35. Button Wiring
36. Nudge & Tilt
37. Plunger
38. Plunger Setup on the PC
39. Fixing VP Plungers
40. Coin Door
41. Audio Systems
42. Backbox Toppers
43. Finishing Touches

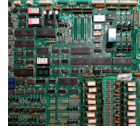
Part Three. *Feedback Devices*

44. Feedback Devices Overview
45. Power Supplies for Feedback
46. DOF Setup
47. Feedback Device Wiring
48. Pinscape Outputs Setup (Expansion Boards)
49. Pinscape Outputs Setup (Standalone KL25Z)
50. LedWiz Setup
51. SainSmart Relay Board Setup
52. LED Resistors
53. Coil Diodes
54. Coil Timers
55. Button Lamps
56. Flashers and Strobes
57. Beacons
58. Undercab Lighting
59. Flippers, Bumpers, and Slingshots
60. Replay Knockers
61. Shaker motors
62. Gear motors



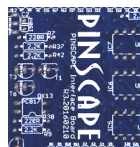
- 63. Fans
- 64. Chimes and Bells
- 65. Addressable Light Strips

Part Four. *A Crash Course in Electronics*



- 66. Electronics Overview
- 67. Static Electricity Precautions
- 68. Circuit board assembly tips
- 69. Field Guide to Components
- 70. Schematics
- 71. Wire
- 72. Resistors
- 73. Capacitors
- 74. Diodes
- 75. LEDs
- 76. Relays
- 77. Transistors
- 78. MOSFETs
- 79. IC Chips
- 80. Connectors
- 81. 0.1" Pin Headers
- 82. Crimp Pins
- 83. Ribbon Cables
- 84. Fuses

Part Five. *The Pinscape Controller*



- 85. Pinscape Controller Overview
- 86. Expansion Boards Overview
- 87. Tools
- 88. KL25Z Hardware Setup
- 89. KL25Z Software Setup
- 90. KL25Z Status Lights
- 91. Electronic Parts List
- 92. European Transistor Substitutions
- 93. Fabricating the Expansion Boards

- 94. Building the Expansion Boards
- 95. Expansion Board Power Cables
- 96. Initial Expansion Board Testing
- 97. Debugging the Expansion Boards
- 98. Connecting the Expansion Boards
- 99. Accelerometer (Nudge) Setup
- 100. Plunger Sensor Breakout Board
- 101. Plunger Setup (TSL1410R Optical Sensor)
- 102. Plunger Setup (Potentiometer)
- 103. Plunger Setup (VCNL4010 Proximity Sensor)
- 104. Plunger Setup (AEDR-8300 Encoder)
- 105. Plunger Setup (TCD1103)
- 106. Plunger Setup (VL6180X Distance Sensor)
- 107. Pinscape Plunger Calibration Button
- 108. Pinscape Config Tool Plunger Setup
- 109. ZB Launch Ball
- 110. Pinscape Button Inputs
- 111. Pinscape Feedback Outputs
- 112. IR Remote Control
- 113. Pinscape Night Mode
- 114. TV ON Switch
- 115. Troubleshooting

Appendices

- Appendix 1. KL25Z Pin Out
- Appendix 2. VP Customizations Summary
- Appendix 3. Plunger Sensor Technical Notes
- Appendix 4. Tables with MagnaSave Buttons
- Appendix 5. DOF Config Tool Device Descriptions
- Appendix 6. DOF Event Codes
- Appendix 7. Customizing a table's DOF effects
- Appendix 8. Coin Door Interface Board
- Appendix 9. Plywood Cutting Plans for Cabinet Construction
- Appendix 10. Cabinet Construction Quick Reference



Appendix 11. A Few Woodworking Tips

Appendix 12. Lock Miter I: The Plywood-Friendly Way

Appendix 13. Lock Miter II: The Special Router Bit Way

Appendix 14. How to Make Corner Braces (and other wood prism shapes)

Appendix 15. A DIY shaker motor plan

End Notes

Part One. *Preliminaries*



1. Preface

I first started thinking about building a virtual pinball machine at least a couple of years before I actually did anything about it. I was on the fence for so long because I thought it was one of those ideas that sounds better than it really is. I also thought I might get bored of it quickly. I loved the *idea* of a life-sized pinball simulator, but I didn't think the real thing could live up to my mental image of it. It didn't help that Visual Pinball always seemed a little disappointing on my desktop PC. I thought a cabinet would just make the video game's limitations more obvious by putting it on a bigger screen. You've probably heard it said that the great thing about a virtual cab is that it's like having 1,000 pinball machines packed into the space of just one, but surely the variety is only of secondary importance: if it weren't fun to play one virtual table, why would you want 1,000 of them?

But I kept coming back to the idea. I'd check in on the forums from time to time to see what was new. At some point, people started talking about putting "feedback toys" in their cabs¹. That's when I finally decided I had to build one. "Toys" are physical devices that create special effects in sync with the on-screen action. The thing that grabbed my attention most was the idea of using solenoids for a tactile *thunk* when a flipper or bumper fires. The ability to *feel* the game action struck me as a whole new dimension that you can't get in mere video pinball.

It's probably needless to say that all of my early reservations were turned around once I started building my cab.

If you're reading this guide, you're probably at least curious about building your own cab. In case you're still undecided, like I was, let me offer this nudge: I think a cab really is a different experience from playing pinball simulations on a desktop PC. It's not just the same thing in a different box. The real controls and the full-scale physical setup are more than just decorations; they're transformative.

I should temper that by adding that a virtual cab won't trick you into thinking you're playing a real pinball machine. It's not *that* realistic. But it's not just desktop video pinball in a fancy box, either. "Virtual" and "real" pinball each have their own advantages, and they're each fun to play in their own way. I'm fortunate to have a small collection of real pinball machines at home, and while I'd never consider the virtual cab to be a replacement for any of those, it is a great addition to the lineup, adding its own unique play style.

Pin cabs aren't just fun to play; they're fun to build. They make great DIY projects. As much as I enjoy playing games on my cab, I also really enjoyed building it, and I'm still coming up with ways to improve it. Most of that work is on the software side these days, although I still tinker with the hardware, too. One of the great things about this hobby is that most of the software involved is open source, so if there's something you don't like, you can change it. That's one of my motivations for sharing the Pinscape Controller project: I wanted to bring the benefits of open source to the hardware side. When I started my cab, most of the hardware options were proprietary devices designed more with video arcade projects in mind, so I'm glad that I've been able to add some more pinball-oriented designs into the mix.

This is a great time to be building a DIY pin cab, thanks to a confluence of trends. One is that real pinball continues to thrive among collectors, which is good for us because it keeps an active market going for the specialized pinball-machine parts we need for our projects. Another helpful trend is the popularity of hobby robotics, which has made lots of advanced electronics accessible to DIYers. A third is the growing interest in virtual pinball itself, which has attracted a talented group of people who work on the open-source software that makes virtual cabs possible. Virtual cabs will keep getting better as long as people are actively working on the

software.

The Pinscape Controller

This guide has grown into a pretty comprehensive set of instructions for building a pin cab, from the woodworking and trim hardware to the electronics and the software. It started out with a much smaller scope, of serving as the user manual for my Pinscape Controller project, and that remains a significant part of the book. The Pinscape Controller is an open-source hardware and software project that can act as the central hub for connecting most of the specialized input and output electronics unique to virtual pin cabs: buttons, sensors, and feedback devices like solenoids and lights. The controller can handle nearly all of the special I/O functions in a pin cab, including:

- Connecting flipper buttons and other pinball-style buttons to the PC and using them to control the game
- Connecting a physical pinball plunger to the software, via a position sensor that detects its position and motion, so that you can launch the ball the traditional way, with precise control for tricky skill shots
- Using an accelerometer to sense the motion of the cabinet, so that you can nudge the cabinet and get a realistic, proportional response in the simulated game
- Connecting feedback devices to the PC, so that the pinball software can create lighting effects and tactile effects cued to the game action

Physically, the core of the Pinscape project is the Freescale FRDM-KL25Z, a tiny, self-contained computing device about the size of a credit card. It costs about \$15 and comes fully assembled and ready to use. You don't need to know anything about electronics to set it up; you just plug in a USB cable and load some software.

The KL25Z by itself does a lot of the heavy lifting. It has a built-in accelerometer (a good one), which nicely handles nudge sensing. You can connect buttons (like the flipper buttons) directly to the KL25Z, with no more work than running some wires.

Beyond buttons and nudging, the electronics work gets a little more complicated. If you want to hook up a plunger sensor or connect feedback devices (lights, motors, solenoids), you have to buy some electronic components, and do some additional wiring and electronic assembly work. That's all laid out in detail in this guide, and it's very scalable - you can do as much or as little of this as you want, according to your interests and comfort level working with electronics. And you can add new features over time; it's all pretty modular. I've tried to make all of the projects approachable even if you don't already know much about electronics.

For the plunger, several options of varying difficulty are available. The easiest option only requires buying a particular kind of potentiometer (about \$6) and connecting three wires. A more complex but more precise option is what I'd call an "intermediate level" DIY electronics project. All options are documented in detail in this guide with step-by-step instructions.

If you want to connect feedback devices to the KL25Z, that also requires some additional electronics work. There are some simple options involving pre-built circuit boards that you can buy on eBay or Amazon. The more advanced and more full-featured options involve the "Expansion Boards", a set of circuit board plans that you can build yourself. Again, this is all documented in this guide.

The Pinscape Controller is an entirely "open source" project, meaning that all of the

software is free to use, all of the source code and hardware designs are published, and you're free to change and customize them in any way. I'm always happy to integrate any customizations that are generally useful back into the official version so that everyone can benefit from them, but you're also free to make private changes for your own use if you prefer.

Cabinet build guide

Beyond the Pinscape-specific material, this guide includes a lot of information about building virtual pin cabs in general. The goal is to provide a complete instruction manual for the DIY pin cab builder. The guide even includes information on the various commercial products that can fill the same roles as the Pinscape project, for people who aren't interested in DIY for those aspects and prefer something that comes ready-made.

I'm including the general virtual cab building material because I would have really liked something like this when I started on my own cabinet. There's a lot of information on building these machines on the Web, mostly in forums and blogs, but it's really scattered and hard to find and navigate. There is one other full build guide out there that I'm aware of, though: Major Frenchy's Mame in a Box, which offers a big library of video tutorials on pin cab building. If your learning style favors video presentations, or you just want another resource to add to this one, check it out.

I can't quite boil things down to a ready-made kit with a master parts list and easy assembly instructions. There are too many possibilities and variations for that. Every cab is unique, which is exactly as it should be for a DIY enterprise like this. So I'll try to go into as much detail as I can, but in many areas the information is more like advice than outright instructions.

Currency and updates

The first complete edition of this guide appeared in October, 2019 (after about three years of partial versions that were kept online as a work-in-progress). "Complete", in the sense of covering all of the topics I set out to cover at a level of detail that I set out to reach - but not necessarily "finished", in the sense of the last possible word being said on every subject, the text set in stone, never to change again. I still see it as an ongoing project, and I revise and expand it from time to time as I encounter errors and omissions, and to keep up with changes in the virtual pinball world. (Most recently in June 2024, according to the electronic librarian that keeps track of the text.)

Many parts of this guide have an inherently long shelf life, because the "real" pinball machines that virtual cabs are based on tend to look and act much the same year after year. I don't think that's likely to change, either, because at least a part of pinball's market demand comes from nostalgia. The pinball makers are aware of this and know that they can't change things too much before people stop thinking of their products as "pinball". A lot of the basic design of a commercial pinball machine (and thus of a virtual cab) is pretty well anchored in the 1990s. That's good for the longevity of this guide, because it means the parts about cab building don't need to be updated every couple of months to chase a new fad. The same is true of the backgrounder sections on woodworking, soldering, and so forth.

Some things do change rapidly, though, some so quickly that I know there's no hope of keeping any guide up-to-date with them. The things that move at warp speed are primarily related to the core electronics - specifically, the TVs and video displays and the PC hardware specs. All of that tends to undergo a complete revolution every four

to six months. I know there's no hope of keeping a list of "Best Intel Chips of 2024" or "The Sharpest TVs Right Now" up to date, so I don't even try to provide such ephemeral shopping lists. Instead, the relevant sections provide a more general, and hopefully more lasting, idea of what to prioritize when shopping. My hope is that this will help the material remain relevant and useful for at least a little while.

In between those extremes - the Moore's-law churn of consumer electronics on the one hand, and the timeless arts of woodworking and soldering on the other - there's another area that changes at a middling pace: the special software and hardware devices for playing virtual pinball. Visual Pinball, PinMAME, DOF, etc - these are generally open-source projects, or in some cases tiny one-person businesses, that are in active development and that come out with new versions once in an unpredictable while. I confess that I don't track every one of those projects closely enough to know immediately when something I've written about it here needs to be updated, and even if I did, it would still take me a while to catch everything up. So it's best to treat the guide as a secondary source of information for the big software components, and look to the projects themselves, or forum activity from their contributors, for the latest news.

If you encounter any errors, or anything that's out of date, I'd be happy if you pointed it out so I could try to fix it. You can contact me on vpforums.org (my user ID there is **mjr.**)

An explanation of "section incomplete" warnings

I originally started posting this guide in draft form in October 2016, when it was just a skeletal outline. Back then, most of the sections were just placeholders, like this:

***This section is incomplete** and will be expanded when time permits. Material to be added: (Some notes about what I intended to write would go here)*

Those placeholders were there so that I could use the guide as its own outline, and also so that readers would know that I hadn't forgotten about the topic in question.

I finally finished filling in all of the planned material in October 2019, so at that point there were exactly zero of those boxes remaining. You probably won't see any in the current guide - but I can't rule that out entirely. I'm still revising and updating the guide on a regular basis, and occasionally a new topic comes up that's big enough that it will take some time to cover. When that happens, I might resort to adding a few of those boxes back in. If you see any, they mean that there's some new material I intend to add when I get a chance.

¹ In the vernacular of the Web forums, a virtual pinball machine is called a *pin cab*, short for "pinball cabinet", and most often shortened further still to simply *cab*. If you're new to the forums, the name might seem weirdly generic. But it makes sense in the context of the forums, because the "cab" groups are only a small part of a broader community that's interested in computer pinball simulation in general. Everyone in the broader community plays computer-simulated pinball, but mostly using ordinary desktop PCs or video game consoles. The thing that distinguishes the "cab" crowd is that we embed our pinball PCs in these elaborate housings based on the real machines.

2. What is a pin cab?

I'm not sure who created the first virtual pinball cabinet, but the seed of the idea is pretty obvious in hindsight. Someone must have been playing around with a pinball program on their desktop PC, and wondered what it would be like to turn the monitor sideways, to make the layout more like a real pinball table's proportions. Then they thought to lay it down flat, so they could stand over it like a real pinball playfield. And then what about switching to a big flat-panel TV that's the same overall size as a real pinball table? The finishing touch was putting the big TV inside an old cabinet salvaged from a defunct real pinball, right where the playfield used to go, for the full life-sized experience.

At its most basic, that pretty much sums up a virtual pin cab. You take a Windows PC, install Visual Pinball and/or other pinball simulator software, attach a TV in the 40" range as the primary monitor, and put the TV inside a pinball cabinet body where the playfield would normally go. Put another TV in the backbox to display the backglass artwork. Connect some speakers in the backbox to the PC sound card, and you have the full audio/visual simulation.

What makes this so special?

Okay, you're thinking, so a pin cab is just a desktop pinball player in the guise of a real machine. Maybe that sounds *somewhat* interesting: the idea of playing on a full-size playfield is pretty novel if you're used to playing pinball in a cramped little perspective window on a PC monitor. But in the end, isn't it just the same game on a bigger screen? Wouldn't the novelty wear off pretty quickly?

As they say in the infomercials, "but wait, there's more..."

Real flipper buttons! Once we have the displays in a proper cabinet arrangement, we can't just plop a boring old PC keyboard on top and go on batting the ball with the Shift keys like on the desktop. It's a real pin cab, so it needs real flipper buttons.

Yes, you can connect the real buttons to the PC. There's a special device called a key encoder that lets you do this. Key encoders are fairly cheap and easy to set up (and the Pinscape Controller can handle this function, of course). They trick Windows into thinking you're still just pressing regular keyboard keys, so you can go on using the same pinball software.

This makes a positively huge difference in playability. If you've played any desktop pinball or tablet pinball, you probably already have a sense for how awkward the controls are; you've undoubtedly lost a good number of balls from having your fingers stray just a little off the keys at crucial moments. Even so, it'll probably still be a revelation the first time you play virtual pinball with real flipper buttons.

You don't have to (and shouldn't) stop at flipper buttons, by the way. You can hook up all of the standard buttons - the Start button, Magna Save buttons, the coin chutes, even the service buttons inside the coin door that let you access the operator menus.

A real plunger! Desktop pinball games all use a "timed" plunger that pulls back as long as you're pressing a key. Which obviously doesn't even remotely resemble how you interact with the plunger on a real machine. On a cabinet, you can install an actual pinball plunger, and connect it to a sensor that lets the PC read its position. This lets you launch a ball exactly like in a real game. The Pinscape Controller offers this capability, and several commercial options are available as well.

Real nudging! Real pinball games are physical, mechanical systems. The game

action obeys the laws of physics, not the whims of a video game programmer. I think that's why pinball remains interesting to so many people even in an age of impossibly sophisticated video games. Pinball's physicality means you can interact with the game in a very direct and visceral way.

Desktop pinball offers an imitation of nudging that's half-hearted at best, letting you "nudge" by pressing a button. Like the timed plunger, it's nice that they tried, but it's nothing like the real thing.

A pin cab makes it possible to bring back real physical interaction. A virtual cab already has the same heft and feel of a real cab, so you'll find yourself unconsciously nudging it like the real thing, your brain expecting it to influence the ball. But what if you really could influence the ball that way? Good news: you can! There's a device called an accelerometer that can measure exactly the sort of motion you impart to the cabinet by nudging. Accelerometers are cheap and ubiquitous these days, thanks to smart phones. The microcontroller board used in the Pinscape Controller has an accelerometer built in, and the Pinscape software is set up to read the acceleration data and send it to the pinball simulator program. Most of the available commercial plunger kits offer the same capability. The accelerometer doesn't just sense the fact that you nudged, but actually measures the amount of force you applied, and passes that along to the simulator, so that the response in the simulation is proportional to the strength of the physical nudge.

Mechanical feedback! Blinking lights! Once the early cabinet builders mastered all of the above, they started coming up with ways to make the experience even more realistic by adding mechanical feedback - devices that actually move inside the cab to simulate flippers, slingshots, and bumpers.

Real pinball is a tactile experience. The solenoids that whack the ball around are really powerful, and you can feel them shake the cabinet every time they fire. Digitized sound effects just aren't the same. For a more realistic virtual experience, you can put solenoids and other physical devices inside your cab that produce similar tactile effects in the virtual game. The pinball software can trigger these feedback devices in sync with the game play. So when a bumper fires in the simulation, you can actually feel a solenoid fire in your cab. You can likewise add bright LEDs and strobes that flash in sync with game events. This makes for a dramatic light show, much more interesting than just images on a TV.

The current generation of pinball software makes it fairly straightforward to attach an array of mechanical devices and external lights. You need another I/O controller for this capability, known in this case as an "output controller". The Pinscape Controller can handle this task, and there are several commercial options as well. As for the feedback effects, there's a fairly standard set of lights, solenoids, and motors that most people use, but the control systems are so flexible that you hook up almost anything you can think of.

That sounds like a lot of work...

If you're new to the virtual cabinet world and you weren't already acquainted with all of these bells (literally) and whistles, you might be feeling a little intimidated by all of this. You might have been picturing just the basics of the Windows PC in the fancy wooden box, and you might have been thinking in terms of what you could build out of ordinary PC equipment. But a lot of the stuff I just mentioned obviously can't be done with ordinary PC parts alone.

The good news is that everything described above is not only possible, but well understood. It's all been reduced to recipes that you can follow. Lots of people have

built cabs with these features. The software infrastructure that supports all of this is mature, and it's specifically designed with all of these features in mind. There's no need to "hack" anything to trick the software into doing these things; the software already knows all about backbox displays, DMDs, nudging, plungers, and feedback devices. You just need to tell it what you have installed, and it'll take advantage of it to create a great playing experience.

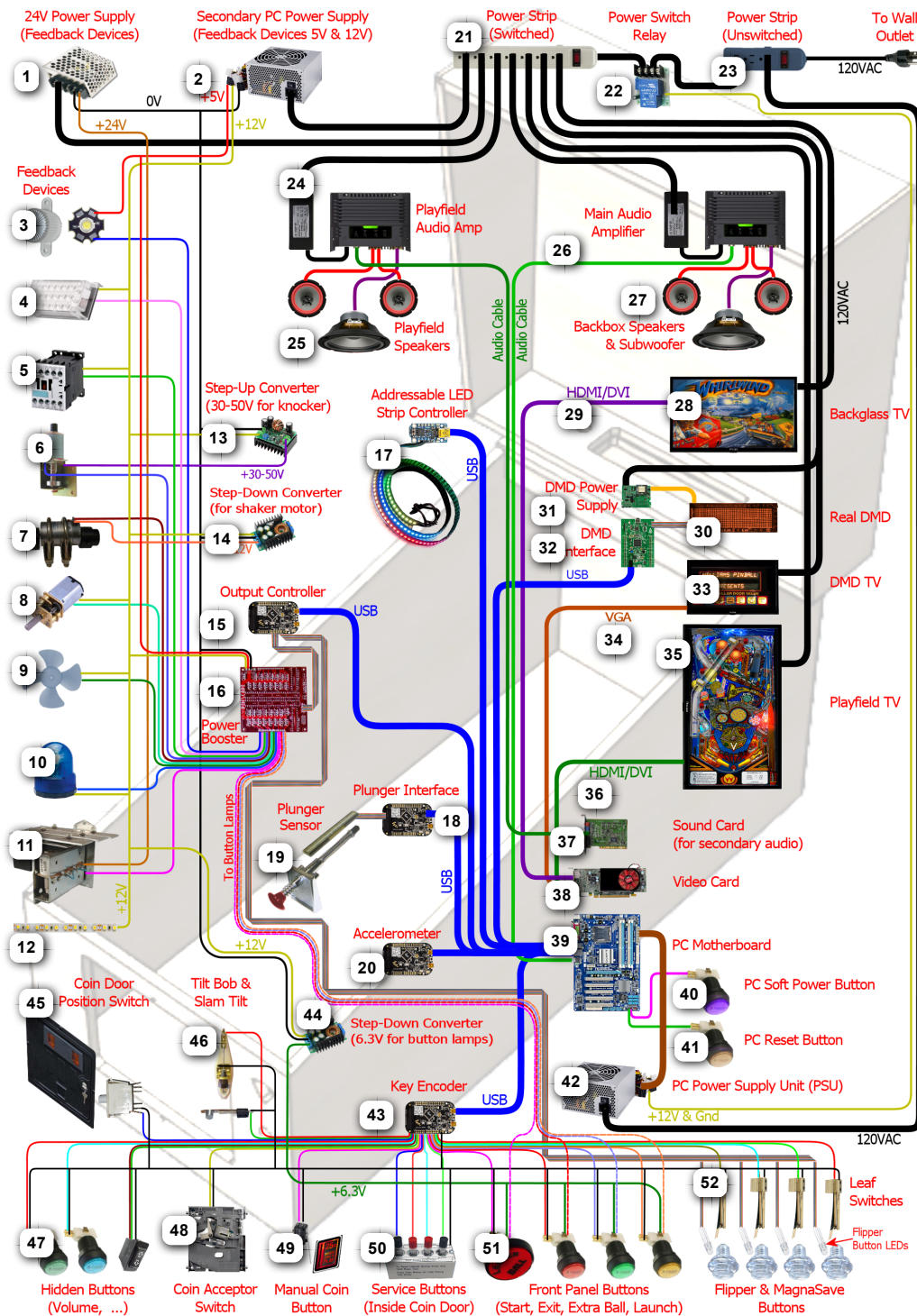
Most importantly, the whole setup can be built by a committed DIYer. That's where this guide comes in. We'll explain how to implement all of the things that go into a virtual pin cab, covering the best DIY solutions as well as the commercial product options.

By the way, "DIY" doesn't mean "poor man's", which is the way a lot of people think of it. In fact, it's really quite the opposite these days, thanks to technologies like ubiquitous computing, 3D printing, and the relentless pace of progress in electronics. DIY means you can build exactly what you want. It means you don't have to settle for what some marketing department thought was good enough for the least common denominator. Yes, you do still see forum discussions about "hacks" and cheap solutions. That's not what this guide is about. The approaches we'll cover in this guide are modern, no-compromises solutions. Some of them have capabilities that you simply can't find in the commercial alternatives.

3. A Visual Guide to the Virtual Pinball Cabinet

When I first started building my own pin cab, I found it hard to get a handle on all of the pieces involved. I knew I wanted real flipper buttons, but how do you connect them to the PC? I wanted a plunger, but how the heck do you make a PC take input from a plunger? I wanted feedback devices to simulate the flipper solenoids and replay knocker and so on, and I understood that I needed an LedWiz for that, but beyond that I didn't really know what it could do or how to connect anything to it. And knowing how many questions I had, I shuddered to think about the questions I didn't even know enough to ask.

The diagram below will, I hope, clear up some of that fog, especially the unknown unknowns. It's is a comprehensive map of all of the electronic systems and devices that make up a virtual pinball machine. It doesn't answer the detailed questions, like exactly how you hook up the LedWiz or install software, but it offers a bird's-eye view of everything that goes into one of these machines. It shows what each component does and how everything fits together. The idea is to let you see the whole system at a glance. This diagram shows pretty much everything electronic in a pin cab, so you should be able to quickly spot any major components that weren't already on your radar, and decide if you need to add them to your plans.



Feature Details

1. 24V power supply

A number of common feedback devices require 24V DC power. These include the most common type of contactors used to simulate flippers, slingshots, and bumpers, as well as real pinball chime coils. If you're using any devices requiring 24V, it's handy to have a dedicated 24V supply in the cabinet, since this isn't one of the voltages you can get from the secondary PC power supply. If you don't have any of the common 24V devices, there's no need for this extra unit.

2. Secondary PC power supply

This is a second PC-type power supply that isn't connect to your PC at all, but is used to provide power to the feedback devices and other devices in your cab. You actually

don't need to use a PC power supply per se here, since we're not using any of its special PC capabilities but just using it as a source of 5V and 12V DC. But PC power supplies are great for this because they're an extremely cheap way to get a good high-current source of 5V and 12V power. The reason we care about having a 5V and 12V supply is that these two voltage levels are exactly what's needed for most of the common feedback devices. And for devices requiring other voltage levels, we can usually use inexpensive step-up and step-down converters that draw their power from the 5V or 12V lines.

3. RGB flashers

Real pinballs usually have a number of "flasher" lights (bright lamps under plastic domes) around the playfield. These produce a brilliant light that's a key part of the show. Emulated tables will of course include the playfield flashers in the on-screen image, but a video monitor can't approach the brightness of the real thing. For a better simulation of the original light show, virtual cabs often include real flashers, usually a set of 5 positioned inside the cabinet just behind the back edge of the playfield TV, and/or on top of the backbox. The flashers in virtual cabs usually use clear plastic domes rather than colored domes, and use RGB LEDs as the lamps, so the software can use them to display flashes in any color. High-power RGB LEDs that produce very bright light can be found on eBay: look for "3W star RGB LED".

Note that this type of LED usually requires a resistor for each color channel, to limit the electric current that flows through the LED element. The resistors aren't shown on the diagram, but the Build Guide section on wiring these devices has full details.

4. Strobe lights

This is another popular lighting effect on virtual cabs. These are simply bright white lights. You can find the type most pin cab people use by searching eBay for "22 LED strobe". These are designed for use on cars and trucks, which means they run on 12VDC, which is perfect for a pin cab.

5. Contactors/solenoids

The most basic "tactile" effect in a virtual cab is the *thunk* made by the flippers, slingshots, knockers, and the like. The best way to implement this is with something physically similar that actually goes *thunk* itself. Digitized sound from the speakers just doesn't have the same tonal quality or create the same sensation in your fingertips. One obvious approach is to use real pinball flipper and bumper assemblies, hidden inside the cab. Some cab builders do go that route, but these are expensive and take up a lot of space. A good compromise that many cab builders use is some kind of self-contained solenoid device. The most popular choice is the Siemens 3RH1140-1BB40 24VDC contactor, which is basically a beefy relay designed to switch high-power loads; it has a solenoid-driven switching mechanism inside that makes a decently pinball-like sound. Another popular option is automotive starter solenoids. Open-frame relays (easily found on eBay) can also serve.

6. Replay knocker

The replay knocker is such a distinctive feature of pinball machines that most virtual cab builders wouldn't dream of being without one. Digitized audio simply can't do justice to the hammer blast of a real knocker. Most cab builders opt for an authentic knocker coil assembly, which you can buy new from any pinball supplier. The real ones aren't very expensive, so hardly anyone bothers to look for a DIY alternative.

7. Shaker motor

If you've ever played a real *Earthshaker*, or almost any recent Stern game, you've probably encountered a shaker motor in a real pinball. Shakers do what the name implies: they shake the machine (and the floor around it) like a minor earthquake is going on. This is one of the most dramatic feedback effects you can include in your machine, and it can really add excitement during play. The implementation is simple:

it's just a DC motor with an off-balance weight attached. It's fairly straightforward and inexpensive to build one yourself. If you do include one, you'll get a lot of use out of it, because the standard DOF setup triggers it at well-chosen events in many tables, even those without shakers originally.

8. Gear motor

Lots of real tables have something on the playfield that's controlled by a little electric motor, which you can hear when it's in action. These motors are usually linked with gears to the playfield features they control, and the gears are what make most of the noise. For a more realistic rendition of this distinctive sound, many cab builders like to include an actual geared motor in their cab. The software can activate this in sync with the game action, just like in the real game.

Two main types of motor are popular among cab builders for this role. The first is the robotics motors available from many sellers on eBay: small DC motors with a couple of brass gears attached. They make about the right amount of noise, but some people think the quality of the noise is too "whiny". The other common type is auto windshield wiper motors. These are usually quieter and lower pitched than the robotics motors.

9. Fan

A small handful of real pinballs featured fans on the backbox, most notably *Whirlwind*. Given the puny number of real machines with fans, you might wonder why you'd want one for a virtual cab. But then you'd be surprised at how many virtual cab builders include one. Like the shaker motor, the standard DOF setup triggers the fan on suitable events in lots of tables that didn't have one in real life, so you'll get a lot more use out of it than just playing *Whirlwind* and *Twister*. And like the shaker, it's a particularly tactile and dramatic effect that can be a lot of fun. Many cab builders use fans that are designed for automotive or boat use (since these conveniently run on 12VDC) and mount them on top of the backbox *a la Whirlwind*. Some cab builders have created sneaky hidden fan setups, with the fan inside the cab, blowing air at the player through an opening in the coin door or a vent under the machine. My own machine uses a replica of the *Whirlwind* fan with a custom enclosure designed to fit the theming of my cabinet art.

10. Beacons

This is another popular light-show effect: a police-car rotating beacon light, or a pair of them, on top of the backbox. A surprising number of real machines had beacons; police themes are apparently as popular in pinball as on CBS. (In fact, *F-14 Tomcat* had **three**, and it wasn't even police-themed.) Like the shaker and fan, you'll get plenty of use out of beacons even in games that never had them originally. It's fairly easy to find suitably sized rotating lights of this sort on-line, especially from auto supply stores. A bonus is that anything designed for cars will conveniently run on 12VDC. There are also novelty beacon lights designed as party decorations, but those tend to be poor imitations.

11. Chimes & bells

If you're a fan of classic electromechanical pinballs, you might consider installing some real chimes and/or bells. As with the knocker, recorded audio doesn't do justice to real percussion instruments. One option is to install an original pinball "chime unit" from the 1960s or 70s. These are basically little xylophones, with three or four metal bars and solenoid-driven hammers. The pinball manufacturers haven't made these in a long time, so if you want the real equipment from the EM era, your best bet is to find a salvaged unit on eBay. It might take some patience since there aren't all that many in circulation. There's also at least one modern reproduction unit available commercially (look for "McCullough's Chime Unit"), but reviews are mixed as to whether it sounds like the originals. The pinball suppliers don't sell full chime units, but they do sell many of the parts, so you might be able to cobble one

together with a mix of original and improvised parts. Another option is to find some suitable "shell" type bells, which were used instead of chimes on many EM machines. You can use a regular knocker coil to strike them. Bells have their own distinctive sound, so a serious EM enthusiast might even want both bells and chimes, for more variety and authenticity in configuring different tables. Yet another possibility is an alarm-clock type of bell that strikes repeatedly when energized, since a few real pinballs had these (*Taxi*, *Space Shuttle*). Finally, a working decorative bell can also make a nice "topper" (something mounted on top of the backbox), as in the original *Fire!*.

12. LED strips/undercab lighting

Many cab builders like to put RGB LED strips on the bottom of the cabinet or the back of the backbox. Look for "5050 LED strip" on eBay; these are adhesive-backed strips with LEDs spaced every couple of centimeters, designed for accent lighting. These strips only show one color at a time across the whole strip (unlike the "addressable" strips, which can show animated effects but require a special controller), so they're best for ambient lighting effects. This is a popular "mod" among real pinball owners, too. It creates a pool of light around the machine, adding to the overall show.

13. Step-up converter for knocker

Real pinball knockers in modern games are mostly designed to run on 50V. They'll work on lower voltages, but to get the proper amount of force for the knocker effect, you need at least 35V or so. There's nothing else in a typical virtual cab that needs that kind of voltage, so in all likelihood you'll end up needing a special power supply just for the knocker. A cheap and easy way to do this is to use a step-up voltage converter. These are available on eBay for about \$15. These take 12V as input (which you can get from your secondary PC power supply), and let you dial in a higher voltage for the output up to some limit. Look for a converter that can go up to at least 35V, and preferably closer to 50V, and one that can supply about 5A at the highest output voltage.

14. Step-down converter for shaker motor

If you build a custom shaker motor, you'll probably want to base it on a 12V DC motor, which means that you could power it directly from the 12V line from your secondary power supply. However, many people find that their shakers are too intense at full power, so they want to be able to turn down the power a bit. One easy way to do this is by dropping the voltage slightly with an adjustable step-down voltage converter. You can find these on eBay for a few dollars. These take 12V as input and produce an output voltage that you can adjust to any lower voltage. Connect the output to the motor, and adjust the output (by turning the control screw on the converter) to get the level of intensity you find most pleasing.

15. Output controller

To control feedback devices, you need an output controller. This is a special device that you connect to the computer with a USB cable. The device takes software commands from the PC and translates them to electrical signals that turn your output devices on and off, in sync with the on-screen game action, to create feedback effects to enhance play. Almost any sort of electrical device can be used for feedback: lights, motors, solenoids, bells, chimes. Some controllers use relays to switch connected devices on and off (e.g., Sainsmart relay boards). Others use solid-state (transistor) circuitry (LedWiz, PacLed, Pinscape). Solid-state controllers almost always require some sort of "power booster" to connect anything more powerful than an LED or small lamp.



The Pinscape Controller can serve as an output controller. If you use the stand-alone KL25Z, the number of outputs is limited: about 20 devices if you're also using it for any button inputs, or about 30 if not. The plain KL25Z also requires

power booster circuits to power anything, even small LEDs. With the expansion boards, the number of outputs is practically unlimited, plus the power boosters are built in, so you can connect powerful devices like solenoids and motors directly.

16. Output power booster

If your output controller has solid-state (transistor) outputs, you'll probably need some kind of power booster to connect anything beyond LEDs and small lamps. The LedWiz and PacLed devices both have this requirement, as does the stand-alone KL25Z with the Pinscape software.

There are several ad hoc solutions that work with any controller. One is to use relays; that's a simple solution, but has some drawbacks. Another is to use the common "LED amplifiers" sold on eBay. These work for high-current LEDs but might not be suitable for solenoids and motors. For a more robust solution, you can use a booster designed specifically for your controller. Zeb's Boards sells specially designed booster boards for the LedWiz and PacLed devices. That's a more expensive option, but easy to set up and superior to the ad hoc solutions.



If you're using the Pinscape Controller with the expansion boards, you won't need any other boosters, because the expansion boards have powerful booster circuits built in. If you're using the stand-alone KL25Z, you can use one of the ad hoc solutions (relays, LED amplifiers), or you can build your own inexpensive solid-state booster circuits using a simple circuit design detailed in the Build Guide.

17. Addressable LED strips

An addressable LED strip is an adhesive-backed strip about a centimeter wide with a row of small LEDs down its length. "Addressable" means that each LED on the strip can be controlled independently, for animated lighting effects. This is still a relatively new and rare "toy" in virtual pin cabs. Some cab builders place these strips along each side of the playfield TV, where the addressable lights can be tied to flashing lights and other events on the playfield.

A special controller device is required to connect this type of light strip to the PC, typically by USB. Free open-source firmware is available that turns a Teensy 3.1 (an inexpensive Arduino-type USB device) into an addressable strip controller that works with Visual Pinball.

18. Plunger interface

This is a USB device that connects the plunger sensor to the PC. Several options are available, including commercial products from Zeb's Boards and VirtuaPin. Many of the available plunger devices also include accelerometers for nudge sensing, and some also include key encoders for button input. Plunger input is sent to the PC as joystick input, since that's the format that Visual Pinball and other emulators use to read the data.



This is one of the roles the Pinscape Controller can fill. In fact, plunger sensing was the whole project's original purpose.

19. Plunger sensor

To connect a standard pinball plunger to the PC, you need some kind of sensor that reads the position of the physical plunger and converts it into an electrical signal. There are many approaches. The Pinscape Controller can use an optical sensor that essentially takes rapid pictures of the plunger and finds the position by scanning the images; it can also work with a potentiometer that's mechanically linked to the plunger, using the varying electrical resistance of the pot to determine the position. The Zeb's Boards kit uses a quadrature sensor, which senses the motion by counting pulses in a moving magnetic bar code. The VirtuaPin kit uses an IR proximity sensor, which uses the brightness of infrared light reflected from the tip of the plunger to

estimate the distance from sensor to tip. The type of sensor you use will depend primarily on which controller you choose; if you go with a commercial kit, it will include the sensor.

Note that virtually all of the sensor options are designed to work with a standard pinball plunger assembly. The commercial kits usually include the plunger. For Pinscape or other DIY options, you can get the plunger assembly from any pinball parts supplier.

20. Accelerometer

Nudging is such an integral and unconscious part of real pinball play that good emulation demands a way to sense when you nudge, shake, or shove the cabinet. The best way to do this is with an accelerometer. A good one can tell the difference between a slight nudge and a hard shove, allowing the simulation to react proportionally.

Visual Pinball and other pinball emulators have good support for accelerometer-based nudging. They take accelerometer input via the standard USB joystick interface, so you just need a device that reports acceleration data this way.



The Pinscape Controller can fill this role (and the accelerometer on the KL25Z is very good). Most of the commercial plunger kits also include a nudge feature, so you probably won't need a separate accelerometer if you have any sort of plunger device. If you decide not to install a plunger, you can install a Pinscape Controller or one of the other plunger kits for its accelerometer features alone if you wish.

21. Switched power strip

This is a second power strip that provides line power to all of the secondary devices in your system: the TVs, the audio amplifiers, and the feedback devices. The line power coming into this strip is controlled by the power switch relay, so the strip receives power when the PC is turned on and is effectively "unplugged" when the PC is off. This provides nice integration for all of the systems in your cabinet so that you can control everything with the main PC soft power button.

An alternative to using the switching relay and a second power strip here is to combine everything into one "smart" power strip designed for computers. A smart strip has a "master" outlet that plugs into the PC, and controls the other outlets according to whether the PC is turned on or off. This is simpler to set up than using a separate relay, but some people have trouble getting these to work reliably. Some motherboards don't seem to draw enough power to trigger the "smart" switching function on some of these strips.

22. Power switch relay

This lets the PC control power to all of the secondary systems in your cabinet: the TVs, the audio amplifiers, and the feedback devices. This works as follows: when the PC is turned on, the 12V power supply from the PC turns on, which activates this relay, supplying power to the second "switched" power strip. When the PC is off, the 12V line turns off, which turns off the relay, which cuts power to the second power strip. This effectively "unplugs" all of the devices on the second power strip. You'll want to choose a relay specifically designed for switching high-power loads. The type designed for air conditioners and water heaters is perfect. Because of the high voltage going through the relay terminals, you'll want to be sure to thoroughly enclose this relay in a protective plastic box so that you don't ever accidentally touch any exposed wires, and so that nothing shaking loose in the cabinet can ever come into contact with the wires.

An alternative to the power switch relay is to use a "smart" power strip designed for computer use. Smart power strips do the same thing as the relay, but this action is

built in to the strip, so you don't have to buy extra parts or do any wiring. Smart power strips are more expensive, though, and some cab builders have had problems with their sensitivity. Smart strips are triggered by the amount of power being drawn through the "master" outlet connected to the PC, so if your PC doesn't draw enough power, it might not trigger the smart strip to turn on the other outlets. An external relay doesn't have this problem because it's triggered by PC power supply voltage output rather than its current input, which makes the relay approach work on every PC. Smart strips can also be perfectly reliable, but this depends on the combination of PC and smart strip model you choose.

23. Main power strip

For the PC power supply connection, you'll want a simple power strip that's left plugged in all the time. This lets you control power to the PC with the "soft" power button. Most people use a power strip with a built-in surge suppressor to protect the PC from power spikes in your house wiring and utility service. For a neat, integrated look to your cabinet, mount this inside the cabinet, and run its power cord out through a hole, to serve as the main power cord that you plug into the wall outlet.

Note that most power strips have a built-in manual switch to turn power to the outlets on and off. Even though we're calling this the "unswitched" power strip, it's perfectly okay to use a power strip with one of these manual switches. You'll just ignore that switch and leave it on all the time.

24. Playfield audio amplifier

This is a **secondary** audio amplifier, connected to your extra sound card. This is an optional system; most cab builders don't bother with it. If you choose to use it, this is simply another amplifier like the one powering your main speakers. This one connects to the second sound card in the PC and to the playfield effects speakers, usually positioned inside the cabinet under the playfield TV. The purpose of this second set of speakers is to audibly place the table sounds effects closer to their simulated sources. The table sounds are things like the ball rolling around and bumping into things, the flippers, the bumpers, and so on - sounds that in the real game would be coming from the playfield area.

One alternative to a separate playfield audio amp is to use the speakers built in to your playfield TV. Many flat panel speakers are too small and tinny for this to sound any good, though. Another alternative is to use a multi-channel amp for the main audio amplifier, with enough independent channels to drive the main speakers **and** the playfield speakers.

25. Playfield speakers

This is an optional, **secondary** set of speakers dedicated to reproducing the playfield sound effects, such as the ball rolling around and bumping into things, the bumpers, the flippers, etc. These speakers should be placed inside the main cabinet, under the playfield. Visual Pinball doesn't currently do anything to position these sound effects spatially, so a single speaker is all you really need here. However, you'll probably want a stereo pair anyway to help spread out the sound so that it doesn't sound like it's all coming from a single point on the playfield. You can also use a separate subwoofer for this set of outputs. Some people use "tactile" subwoofers here - the type that video gamers and home theater enthusiasts attach to their chairs to create a Sensurround® effect. A tactile subwoofer can let you feel the ball rolling and bumping effects through the cabinet, which can add to the realism, although you might find that you have to edit some tables to tone down their effects. Some are too much of a good thing with a tactile sub.

An alternative to using separate speakers here is to play these effects through your playfield TV's built-in speakers. Flat panel TV speakers are often too small and tinny for this to sound any good, though; many table effects need good bass reproduction

to sound right.

26. Main audio amplifier

The audio amplifier for your main speakers. This connects to your PC's audio output - the "line out" jack on your motherboard, if it has one, or on your main sound card - with a standard audio cable. Many types of amplifiers can be used here. Many people use car amps, since they're compact and run on 12V, which means they can be powered from a PC power supply. Another popular option is to use powered computer speakers. You could even use a home-audio receiver or amplifier, although these tend to be too bulky to easily fit in a pin cab. If you use a subwoofer, you'll want at least a "2.1" channel amp - two stereo channels plus a mono subwoofer channel. Some 4-channel car amps can be wired with one pair of channels "bridged" together to serve as the subwoofer channel.

27. Main speakers and subwoofer

If you're building your cab in the style of the real machines from the 1980s and 90s, the main speakers will consist of a pair of small (4" to 5") "satellite" style speakers mounted behind the speaker panel in the backbox, and a separate large (8" to 10") subwoofer mounted on the floor of the main cabinet, facing down through a circular opening. Cab builders often use car speakers for these, since many good options are available in the right size range. If you're building your cab in the style of an older machine from the electromechanical era, you'll have to be more creative about where to put the speakers, since the "sound systems" on those machines consisted of actual noisemakers (chimes and bells), not speakers.

28. Backglass TV

This is a TV positioned where the translite or backglass would normally go in a real pinball. This is connected to the PC video card with HDMI or DVI like an ordinary PC monitor, and Windows sees it as a second display. If you haven't done this before, it's easier than it sounds, because Windows has built-in support for multiple displays that actually works pretty effortlessly. Visual Pinball and other emulators can easily be set up to display the animated backglass graphics on this separate monitor for realistic play.

If you build your cab following the 1990s style, with a separate speaker/DMD panel, most 30" widescreen (16:9) TVs will be a good fit. They're almost exactly the right width, but they're not quite tall enough, so there will be about a 1" gap above and below. You can cover the gap with a painted or decal mask on the translite.

Some cab builders opt for a single monitor filling the whole backbox area rather than using a separate speaker panel. That arrangement is even trickier because the backbox has a nearly square aspect ratio, and square TVs simply don't exist. The usual solution is to use a widescreen monitor in portrait mode, and submerge part of into the cabinet below the backbox. This has disadvantages, obviously.

29. Backglass TV video cable

The backglass TV connects to the PC video card with an ordinary video cable, usually HDMI on the TV side, and either HDMI or DVI-D on the PC side.

30. Real DMD

Most real pinballs from the 1990s and later used Dot Matrix Displays, or DMDs, positioned in the speaker panel at the bottom of the backbox. The real DMDs from the 90s were mostly 128x32 plasma displays; these are extremely bright and have a distinctive amber color. Recent Stern games have switched to LED displays with the same pixel layout, and still in monochrome, but with different colors on different games. Visual Pinball and some other emulators can take advantage of the authentic equipment, either plasma or LED, to display animated graphics just like the real machines. You can't get more authentic for the games that had DMDs originally. For

a modern variation, full-color RGB LED panels are now available with the same pixel layout, allowing more variations than the traditional monochrome. A slight drawback to real DMDs is that their low resolution makes them less flexible for games from the pre-DMD era, such as the alphanumeric games. Another complication is that you'll need some extra hardware: namely, a DMD interface board to connect it to the PC, and in the case of a plasma DMD, a special power supply.

31. Power supply for real DMD

If you're using a **plasma** Dot Matrix Display (DMD), you'll need a special power supply module just for the display, since the plasma panels require high voltages that you can't get from a regular PC power supply. Suitable power supplies are available commercially, or you can build one yourself if you're good with electronics. Most LED DMDs run on 5V, meaning they don't need separate power supplies but can use the regular PC PSU.

32. DMD interface module

If you're using a Dot Matrix Display (DMD), you'll need a special device to connect the DMD to the PC. This applies to both the plasma and LED panels. DMD panels won't work directly with a PC, as they don't have any of the necessary electronics on board to connect to a regular video source. Fortunately, there are special interface modules available that bridge this gap. These connect to the PC via USB cable, and translate the PC software commands to the electronic signals that control the DMD. One option is a commercial product called PinDMD (available in versions, PinDMD2 and PinDMD3). Another option is an open-source DIY project with the confusingly similar name Pin2DMD.

33. DMD TV

Most real pinballs from the 1980s and 90s had score displays positioned in the speaker panel the bottom of the backbox. The early versions of these panels used 14-segment alphanumeric displays. More modern games changed to Dot Matrix Displays (DMDs), which can display full graphics, albeit at fairly low resolution (usually 128x32 pixels). One way to simulate both types of display is to use a small TV or a laptop LCD panel, positioned in the speaker panel where the DMD would go in a modern machine. Like the playfield and backbox TVs, this is just another video monitor as far as Windows is concerned. Visual Pinball and other pinball programs can take advantage of it show the DMD graphics or alphanumeric score. A 15" laptop screen is almost exactly the right width for the standard DMD size of real pinballs; it's taller than the real thing, but you can hide the excess height behind the speaker panel. An alternative is to use a real pinball DMD. Another is to leave this out entirely, and overlay the DMD area onto the bottom of the main backglass TV.

34. DMD monitor video cable

If you use a TV or video monitor for the DMD area, this connects to the PC video card with an ordinary video cable. This is usually VGA for a third monitor, for the simple practical reason that most video cards don't have three HDMI/DVI ports but do usually have a spare VGA port left over even after connecting two other monitors.

35. Playfield TV

A large TV or monitor goes where the main playfield sits in a real pinball. You connect this to the PC video card with an ordinary video cable, and Windows simply sees the TV as a monitor. A regular 16:9 widescreen TV is a pretty good approximation to the aspect ratio of a real playfield when rotated 90° (for "portrait mode"). You can either choose a TV that fits your cabinet, or build a custom cabinet around your TV. Before deciding, you should be aware that all of the pinball cabinet hardware you can buy off-the-shelf is designed to fit just two size options, known as "standard body" and "widebody". If you build a cabinet with custom dimensions, you'll need custom versions of some of the accessory hardware. The standard body is 20.5" wide on the inside, which is enough to fit most 39" TVs and some 40". The

widebodies are 23.25" wide on the inside, which will fit up to about a 45" TV. In terms of the displayed image size, a 39" TV yields about the closest match to the true object sizes; the image on a larger TV in a widebody is a bit larger than life. Many people use widebody plans anyway, for the greater flexibility in choosing a TV, and also because larger-than-life can also be fun.

36. Playfield TV video cable

The playfield TV connects to the PC video card with an ordinary video cable, usually HDMI on the TV side, and either HDMI or DVI-D on the PC side.

37. Sound card for secondary audio

Visual Pinball can take advantage of **two separate audio systems**. The first is used to play back the original "soundtrack" of the game (the music, speech, and sound effects that played through the backbox speakers on the original arcade machine). The second system plays the "table" sound effects, such as the sound of the ball rolling around the field and hitting things, and the sounds made by the bumpers, flippers, and other solenoids. It makes the simulation a little more realistic to play the table effects from speakers inside the cabinet, under the main TV, closer to where they'd come from in a real machine. To take full advantage of VP's ability to separate the sound effects, you have to add a separate sound card. Most modern motherboards have a "sound card" built in, so all you need is one add-in sound card to get the second set of channels. This might sound like it's asking for trouble with Windows device conflicts, but it's actually no problem, as Windows has good support for using multiple sound cards. Connect your backbox speakers to the primary audio output (usually the one on your motherboard), and connect your in-cabinet "table effects" speakers to the extra audio card. The second sound card is completely optional, as VP will play everything through a single set of speakers by default, but the extra spatial separation from a second set of speakers is a nice little enhancement.

38. Video (graphics) card

Pinball emulators are fundamentally video games, so a good video card is important. Video pinball doesn't lean on the graphics processor quite as heavily as the most demanding 3D games, so you don't need a super high-end gaming rig, but you'll definitely want something more powerful than a basic business-graphics card. Look for a good mid-range gaming card. An important feature to consider is support for multiple monitors. If you plan a 3-monitor setup (playfield, backglass, DMD), be sure your card has at least three outputs, with a set of connectors compatible with your monitors. The reason that multiple-monitor support is important is that most people find that you get much better performance by connecting **all** monitors to **one** video card than splitting monitors across cards.

39. PC motherboard

The heart of a virtual pinball machine is a standard PC motherboard running Windows.

40. PC soft power button

Standard PC motherboards have wiring for connecting a "soft" power button - push the button to turn the PC on, push it again to tell Windows to power down. The button wiring can be connected to any ordinary momentary pushbutton switch. The type of button commonly used for a real pinball machine's front-panel Start button is a good choice, because it's easy to install in cabinet and has an integrated microswitch that's easy to wire. On real pinball machines, it's standard to place a "hard" on/off switch (which physically connects and disconnects line power) on the bottom of the cabinet, near the right edge and a few inches back from the front. This is nicely hidden away but easy to reach and easy to find by feel. For a virtual machine, I recommend placing the "soft" power button in the same spot.

41. PC reset button

Most PC motherboards have wiring for connecting a reset button, to forcibly reboot the system in case the operating system crashes. This can be connected to a simple pushbutton switch, such as the type used for the soft power button or the front-panel Start button; one possibility is to place this on the bottom of the cabinet near the power button. Or you can connect this to something akin to the "service buttons" inside the coin door. For modern Windows systems, this type of button isn't all that useful, but some people like to include one just in case.

42. PC power supply

An ordinary PC power supply unit (PSU) is needed to power the motherboard. This should be connected to an **unswitched** power inlet, since the PC should always be physically plugged in to wall power to allow turning it on with the soft power switch. I recommend powering **only** the PC components with this PSU, and using separate power supplies for the feedback devices.

43. Key encoder

This is a core device that almost every virtual pinball machine needs. It lets you to connect real pinball buttons (flipper buttons, Start buttons, etc) to the PC. Most of these devices connect via USB, while some connect to a PS/2 keyboard port. Depending on the device, the physical buttons in your cabinet are mapped either to keyboard input on the PC or joystick button input. Some key encoders let you program which keyboard keys or joystick buttons are sent to the PC, and some have pre-set mappings that you can't change.



The Pinscape Controller can fill this role. It lets you map buttons to keyboard keys or joystick buttons of your choosing (or a mix of the two), and lets you program all of the mappings individually. Pinscape also lets you assign a "Shift" button that gives every other button a secondary assignment, letting you access more functions without adding more physical buttons. There are also commercial key encoder devices available that offer similar features, including the i-Pac and KeyWiz. Or, if you buy one of the commercial plunger kits, it will probably provide button input as a bonus feature, although it'll have a limited number of inputs and probably won't let you choose your own key mappings.

44. 6.3V step-down converter

The most common type of illuminated pushbutton for the front panel of your machine (e.g., the Start button) uses #555 light bulbs. These bulbs are designed to run on 6.3V, which is a rather odd voltage that you won't find anywhere in a PC. These bulbs will also work on 5V (available from the PC power supply), but they'll look a little dim at the reduced voltage. If you don't like that, one solution is to replace your incandescent #555 bulbs with LED equivalents, most of which will work on 5V without loss of brightness. Another solution is to keep the incandescent bulbs and supply them with the right voltage, by using an adjustable step-down voltage converter, which can be found on eBay for a few dollars. What these do is take a power supply voltage on their input terminal, say 12V, and let you select a lower voltage on the output terminal by turning a dial. Get one of these and set it to 6.3V for your illuminated buttons. A single converter can supply power to multiple buttons.

45. Coin door position switch

Real pinball machines have a switch that senses when the coin door is open. This is usually implemented with a plunger switch that's pushed in by a bracket when the coin door is closed, and released when the door is open. You can get the authentic type of switch from pinball suppliers. You can also use a regular microswitch, although it's a little harder to get the mounting geometry right with such a small switch. You can also just install a manual "coin door" button, which is simpler to set up but a little less convenient to use, obviously. In any case, it's useful to have

something to serve this role, since many tables won't let you access the service menu unless they think the door is open, which requires that they get the appropriate switch signal. Whatever type of control you choose for this, you can connect it to the key encoder like your other buttons.

46. Tilt bob & slam tilt

If you have an accelerometer, you'll probably also want a real tilt bob. This is a really simple device that consists of a freely hanging metal weight surrounded by a metal ring. When the weight touches the ring, it makes electrical contact and acts like a switch. Shaking the machine makes the weight swing; shaking too much makes it swing far enough to touch the ring. The pinball software can simulate this at a simplistic level using the accelerometer data, but real cabinet motion is complex enough that the simulation isn't usually very convincing. A real tilt bob works better, and it's cheap enough and easy enough to set up that I think every cab with an accelerometer should have one. Just wire it to the key encoder like a button.

There's another type of tilt detector called a "slam tilt". It's usually built in to real coin doors. It looks like an oversized leaf switch with a big metal weight at the end of one leaf. This detects hard shoves on the front of the cabinet, mostly to deter arcade customers from seriously abusing the equipment or trying to break into the coin box. It's not very important in a virtual machine because you're probably going to treat it more kindly anyway. But if you're a completist, you can connect this to the key encoder like your other switches. As with other coin door items, it will probably be wired to the coin door wiring harness.

47. Hidden buttons

Some cabinet builders like to add a few extra buttons that aren't part of a real pinball, but serve some special "virtual" function. And because the buttons aren't authentic, many builders like to put such buttons somewhere out of plain sight, so as not to affect the aesthetics. One good hidden location is the bottom of the cabinet, near the front edge, where buttons can be easily reached and identified by feel. This is good for buttons you might want to access frequently, like a volume control or a manual "Coin In" button. Another option is to hide buttons inside the cabinet, close to the coin door or even mounted on the coin door itself. This option is best for buttons that you'll access infrequently or that you don't want curious guests messing around with, like a manual "coin door" button or service menu buttons.

48. Coin acceptor switch

If you have a real coin door, and you choose to install real coin acceptors (often called "coin mechanisms" or just "mechs"), you can set up your machine so that feeding in a quarter sends a coin signal to Visual Pinball. This is actually pretty easy, because the coin mechs use a simple microswitch that's tripped by the passage of a coin through the acceptor slot. So all you have to do is wire the switch to your key encoder. The slight complication (as with the coin door service buttons) is that the coin switches are usually wired to a connector or wire harness along with all of the other coin door wires, so you might have to spend a little time sleuthing out which wires connect to the coin switches. Once you do, just connect them to the key encoder. Note that if you also have a manual "Coin In" button, you can simplify things by wiring the coin acceptor switch and Coin In button in parallel, so that either one can be used to add a coin in the simulation.

49. Manual coin button

It's handy to have a dedicated button somewhere on your machine to simulate inserting a coin. You won't need it very often, because it's fairly easy to set most games to Free Play mode where coins aren't needed. Some older games are difficult or impossible to set to Free Play, though; the easiest way to handle them is to feed them fake coins with a button. Some cab builders just add a Coin button to the front panel, alongside the Start and Exit buttons. Others hide a button under the bottom

of the cabinet or inside the coin door. My favorite solution is to use the Coin Reject buttons on the coin chutes, assuming you have a standard coin door. You can position microswitches behind these buttons so that pushing one triggers a switch. Whatever placement you choose, you can simply wire this button to your key encoder like the rest.

50. Service buttons

Real pinball machines from the 1980s onward have a set of "service" buttons inside the coin door. These let the operator access the machine's setup menus for adjusting game options, pricing, etc. The same buttons are useful in your virtual machine because they let you make the same types of adjustments to the virtual tables you play.

If you have a real coin door on your machine, it'll come with the standard set of service buttons for its vintage (older machines usually had three buttons, newer machines usually have four). You can wire these to your key encoder just like all of the other buttons. The only complication is that your real coin door will probably be pre-wired to some kind of wiring harness or connector, so you'll have to figure out what each connector terminal is wired to. An alternative to using the genuine coin door service buttons is to install some extra buttons of your own, probably placed out of sight somewhere (see "hidden buttons").

51. Front panel buttons

Most virtual pinball machine builders include the most common buttons that real machines have. The only one that's truly universal is the Start button, but enough machines from the 1990s also had an "Extra Ball" button (sometimes called "Buy In" or something else) that many virtual cabs include one. A "Launch Ball" button is also useful, as a fair number of 1990s pinballs had buttons in lieu of plungers. Some cab builders even dispense with the plunger entirely and use only a Launch button, since plungers are more expensive and more complicated to set up than buttons.

Some cab builders add one or more buttons on top of the lockdown bar (the bar at the front that holds the top glass in place), replicating the "Fire" button found on many recent Stern games. A handful of older games had one or more extra buttons here as well. These are trickier to install than front-panel buttons because they need holes in the lock bar.

Virtual pin cabs need at least one extra button that real games lack: an "Exit" or "Menu" button, to exit the current table and return to the game picker menu. Some cab builders add other special-purpose buttons for other game picker functions, such as "Instructions" or "Flyer"; others prefer to keep the front panel buttons to a minimum, to be more like real machines. A "Coin In" button can also be handy, although this can be handled more elegantly by using the Coin Reject button on the coin chute.

The best choice for most of the front panel buttons is the "button & lamp assembly" that you can find from pinball and arcade suppliers. These have everything in one nicely integrated package: pushbutton, trim, microswitch, and lamp. They easily mount on the panel face through a round drill hole, and can optionally be set flush with the panel by routing a recess. ("Launch Ball" buttons have a larger button face, but these are available in the same type of combined assembly.) The buttons have four terminals on the back: two for the switch, and two for the lamp. Connect the switch terminals to your key encoder, and connect the lamp terminals to your output controller. The output controller can then turn the button light on at appropriate times in the game.

52. Flipper buttons

If there's a single must-have feature for a virtual pinball machine, this would be it.

There's a certain feel unique to real flipper buttons, which makes them essential to proper emulation. You can order standard flipper buttons from a pinball supplier, along with leaf switches. The leaf switches connect to your Key Encoder the same way as any other switch.

Most cab builders install *two* sets of flipper buttons on each side of the cabinet. The second set is optional, but I'd recommend them. The front set, as you'd expect, is for the flippers. The rear set, located just behind these, is used for other functions that vary by table. Everyone calls these "Magna Save" buttons, because that's the most widely known feature on real machines that used extra buttons like these. But it's a bit of a misnomer, since most of the tables that had extra buttons like these used them for different purposes entirely. There are enough tables that take advantage of extra buttons that most cab builders think it's worthwhile to include them.

Starting in the mid 1990s, the real pinball manufacturers began moving from leaf switches to optical-interruptor switches. You can use optical switches in a virtual cab if you wish, although they're more expensive, and they don't feel exactly the same. The main reason they moved to optical switches on real machines is that they stand up better to heavy arcade use than leaf switches. That's less of a concern with home use, so most cab builders go with the cheaper and simpler leaf switches.

Flipper buttons are available in numerous colors, so you can choose something that coordinates with your cabinet artwork. Or, for a cool lighting effect, use clear buttons, and place a pair of small RGB LEDs behind each one. Connect the LEDs to your output controller. The pinball software will then be able to light up your flipper buttons in the same color originally used on the real machine whenever you load a table.

Notes

- The machine shown here is fully decked out. A bit more than fully, in fact: some things are redundant, such having both a "real DMD" and a "DMD TV". If you're in the planning stages for building a cab, you'll only need to consider the components related to the features you plan to include.
- This isn't a complete wiring diagram or schematic. For that, refer to the Build Guide sections on the individual subsystems.
- "DOF" (referred to several times in the detail popups) stands for DirectOutput Framework, one of the key pieces of software you'll want to install on the PC inside a cab. DOF is the software that handles the feedback devices.

How the Pinscape Controller fits in

You won't find any one box on the diagram that represents the Pinscape Controller. That's because Pinscape is only one of several possible choices for the functions it performs, and because Pinscape can perform several different functions. So the diagram instead shows boxes for the individual functions conceptually. If you do decide to use a Pinscape Controller, it can fill any or all of these roles:

- Key encoder
- Accelerometer
- Plunger interface
- Output controller

Even though these functions are shown as separate boxes on the diagram, **a single Pinscape unit** can fill **all** of these roles simultaneously.

The Pinscape Expansion Boards also serve as the "Power Booster", which is shown as another separate box. If you're using the stand-alone KL25Z without the expansion boards, you'll need something to serve as the power booster if you want to connect feedback devices. The Build Guide includes circuit plans.

4. Resources

Here are some resources I've found helpful while building my cabinet.

Contacting the author

I encourage you to use the virtual cabinet forums for most support-type questions. The forums have the advantage that other people might be able to answer before I see the question, so you might get an answer more quickly. The other plus is that the public discussion might help other people who have similar questions.

If you don't want to post the question publicly for some reason, you can send me a private messages (PM) on vpforums. My ID there is **mjr**.

Other build guides

- Major Frenchy's Mame in a Box offers a large set of video tutorials on building pin cabs. Its goal is to be a comprehensive build guide like this one, including a lot of material in video format.
- Pinball Electrical 101 by Maxxsinner. This is one of the first guides to wiring common I/O devices in a pin cab, primarily buttons and feedback devices. This guide is quite old and out of date at this point, and many of its ideas have been superseded by better and more modern ways of doing things, so I wouldn't use it as a blueprint for a new build. I'm mentioning it mostly for historical reference, since it was highly influential in the early days and had some great ideas that led us to where we are now.

Web forums

There are a number of Web forums dedicated to virtual cab building. These are good places to seek help with general questions on your build, and to connect with other virtual pinball enthusiasts.

- VP Forums > Virtual Pinball Cabinets forum. I read this one regularly and try to help out with questions when I can (Pinscape-related and otherwise). This is the most active of the cab-focused forums I frequent. It has a broad membership, a fair number of whom are long-time members with a lot of pin cab experience. Technical questions posted here usually get prompt and helpful replies.
- VP Universe > Cabinet Discussion. This is another good cab builder forum. It has more of a DIY bent than VP Forums, which you'd think would make it a better fit for my projects, but I have to admit that I don't keep up with this one as much as VPF.
- Pinside. This is a forum site for people who collect *real* pinball machines, so don't go here for help setting up Visual Pinball or an electronic plunger. Even so, the real pinball sites can still be interesting to visit, because virtual cabs do have *some* things in common with the real ones, particularly in the physical construction of the cabinet. Pinside can also be a great resource if you're working on a Visual Pinball re-creation of a particular table. It's a good bet that you can connect there with collectors who own the table you're working on, and they might be able to answer questions about the finer points of that game's operation, send you closeup photos of playfield parts, etc.

Pinball cabinet bodies, pre-built and kits

- VirtuaPin sells everything from individual replacement parts to complete, fully assembled, fully operational virtual pinball machines. They have an especially good selection of parts and kits for DIYers. Notably, they sell "flat pack" cabinet kits, with all of the wood for a cab pre-cut, routed, mitered, and ready to assemble: kind of like an Ikea bookshelf without the ümlauts. They also offer fully assembled (empty) cabinets. They use quality materials and computer-controlled cutting, and let you customize the standard plans if you have something unusual in mind (e.g., non-standard dimensions to fit a specific monitor). Their standard cab kits are faithful replicas of the Williams/Bally 1990s cabinet design, which in my opinion is the gold standard. I used a VirtuaPin flat pack as the basis for my own machine and couldn't be happier with it.

Pinball cabinet artwork - design

- Stuzza on VP Forums has long been designing custom cabinet artwork for other forum members. You can commission a custom project through him for a fee. He doesn't do the printing, just the graphic design work, providing the artwork in digital form ready for printing. He also has a large collection of past work that he makes available for free.
- VirtuaPin offers custom graphic design services for a fee. They also offer several ready-made themes of their own design, as well as licensed reproductions of the original artwork from several real pinball machines.

Pinball cabinet artwork - printing

- Brad Bowman a/k/a Lucian045 on VP Universe offers custom decal printing for pin cabs. Brad has a unique combination of skills for this, because he not only runs a print shop, but also happens to be a long-time virtual cab enthusiast himself. Brad offers special pricing to pin cab builders in his VP Universe thread. Brad printed the decals for my own machine; he was great to work with and the print work was top notch. I was also greatly impressed by the materials he uses. I read a lot of horror stories before building my cab about how difficult it is to install decals properly, but the media that Brad uses are easy to work with. Brad's standard package is for the main cabinet and backbox, but he was able to do my speaker panel and translite art as custom add-ons.
- VirtuaPin also offers custom decal printing. VirtuaPin's standard package is for the main cabinet and backbox art, and they have add-on options for speaker panels and translites.
- GameOnGrafix.com offers custom decal printing for arcade machines. They specialize mostly in video game cabs, but they have a pinball package as well. Look for it under the "customer designed" section.

Pinball cabinet hardware and other pinball parts

- VirtuaPin stocks most of the essential cabinet hardware used in virtual cabs. They have some nice bundle kits that are good deals and can save you a lot of time figuring out which parts you need. Their selection is limited, but they have good coverage for the most common virtual cab parts.
- Pinball Life is a parts supplier for real pinball owners, which makes them a

great place to find authentic cabinet parts. They sell a wide range of replacement parts for the real machines. I've ordered from them several times (both for my real machines and my virtual cab) and have always had good experiences. PBL's catalog is a good middle ground between VirtuaPin's very targeted selection and Marco's almost too-big catalog.

- Marco Specialties is another pinball parts supplier. I've ordered from Marco a few times and recommend them. They have by far the biggest catalog of all of the suppliers I've encountered. So big that it's almost too big! It's so big that it can be difficult to find what you're looking for. But by the same token, if you need something obscure, this is often the only place to find it. Tip: if a search on the Marco site turns up so many hits that your eyes are glazing over scrolling through the results, try a Google search for **Marco Specialties** plus what you're looking for. Google tends to be better at finding the most relevant matches, so it might get you to the right product page quicker. Also, if possible, search by part number, since more generic keyword searches can turn up so many hits, and since the Marco product listings don't always use the same keywords you're expecting.
- Planetary Pinball Supply, yet another pinball parts vendor. PPS mostly stocks specialty parts for machines from the 1990s and later, with better coverage for later models. They don't seem to have nearly the same catalog depth as Marco, but what they have in stock often seems to be somewhat complementary to Marco: if it's a reasonably common part and Marco doesn't have it on hand, try PPS, and vice versa.
- SuzoHapp is a giant manufacturer of parts for all things coin-operated, which incidentally includes pinball machines. They're the original manufacturer for a lot of the standard parts that Williams and Stern used. They make a lot of the basic parts you find in common across machines, such as buttons, trim, coin doors, and so on. The pinball suppliers above resell a lot of SuzoHapp's pinball-related parts, but it can be worth checking SuzoHapp's site as well, because they sometimes offer additional variations of the parts that the pinball supplier sites don't sell.
- Bolt Depot is a great source for fasteners. If you haven't built a pin cab before, you're probably thinking that nuts and bolts are just an afterthought and that you can pick up what you need at Home Depot. But pin cabs actually use a lot of weird specialty fasteners that can be tough to find. There's nothing pinball-specific about Bolt Depot, but I'm including them in the Cabinet Hardware section because they turned out to be a great source for most of the obscure nuts and bolts I couldn't find at my local big-box stores.
- eBay is great for some things, but it's definitely not the first place I'd look for pinball parts. The main reason is price. There seem to be a lot of price gougers on eBay when it comes to pinball parts. I've seen many cases where the price for a shiny, brand-new part from Pinball Life or Marco Specialties is less than the eBay price for the same part in a rusty, beat-up, old used version. Even so, some virtual cab builders have gotten lucky with great deals on eBay, so it can be worth doing some comparison shopping there as you compile your parts lists. eBay is a good option - sometimes the only option - for out-of-production parts that the suppliers no longer carry. For example, chime units from the 1970s are all but impossible to find anywhere else.

Special-purpose electronics for virtual pinball

Given how obscure a hobby this is, it's kind of amazing how many commercial products are available for it. Here are some of the specialized products that cab

builders often find useful.

- Zeb's Boards makes a number of electronic devices specifically for virtual pin cabs, ranging from add-ons and accessories for DIYers (LedWiz booster board, voltage converters) to complete turn-key solutions for input and output (plunger kit, feedback kit). Zeb's is well regarded for great products and excellent customer service. (I'm a delighted customer of Zeb's myself.)
- VirtuaPin sells a number of specialized pin cab devices, including a plunger kit and DMD (dot matrix display) kits.
- Groovy Game Gear makes the LedWiz, which is probably the most widely used device in pin cabs for connecting feedback devices. They also make a key encoder device, and sell lots of arcade game accessories like joysticks and buttons. Their focus is actually DIY video arcade games rather than pinballs, but there's obviously a lot of overlap between the two.
- Ultimarc is another company that makes products for DIY video games that can cross over to virtual pinball. Notable Ultimarc products include the PacLed output controller devices (similar to the LedWiz) and the i-Pac key encoder. They also sell controls like joysticks and buttons.
- Arnoz sells a number of fully built circuit boards based loosely on the Pinscape Expansion Boards designs. His system is modular, so you can buy what you need and add onto it as you go. This is a great option if you want some of the Expansion Board features but you don't want to build the boards yourself.

TVs and other general consumer electronics

I probably don't need to mention any of these unless you time-traveled here from 1972, but then again, maybe you did; pinball *is* an anachronistic sort of hobby... Very briefly, a few places to look for your cabinet TVs and other basic consumer electronics:

- Amazon.com
- Best Buy
- NewEgg
- Tiger Direct
- eBay, especially if you're looking for something used

PC components

As with the consumer electronics, you probably already know the right places to go for PC components. But for the sake of completeness:

- NewEgg
- Tiger Direct
- eBay
- ioFast is a good source for cables of all sorts (computer, network, video, audio) at bargain prices
- Monoprice is another good place to find cables

Electronic parts and components

- Mouser Electronics is my go-to Web retailer for electronics. Mouser is a major electronics distributor that carries a staggering range of components. They have just about everything electronic you could ever need, and their product pages have excellent technical detail and links to manufacturer data sheets and documentation. Their prices are moderate: generally lower than buying the same thing from a general retailer like Amazon (when you can find it there at all), but generally higher than buying from the cheapest sellers on eBay (again, if you can even find it there). One of Mouser's great features is their careful packaging for loose parts: everything gets wrapped in clearly labeled zip-lock bags, so you can easily tell the 100-ohm resistors from the 1K resistors without having to read the color-stripe codes. Mouser's only downside is also one of their big virtues: that their catalog is so huge. The vast number of parts they stock can make it hard to find things, even with their excellent parametric search system. Fortunately, you won't have to do your own searches for most of the Pinscape parts, since we give you exact part numbers for just about everything.
- DigiKey is another major electronics distributor very much like Mouser. They have a similar selection and similar prices.
- Newark is yet another distributor. They also run the element14 community for electrical engineers and hobbyists, which has forums and online articles related to electronics.
- eBay is a good place to find *some* electronics. Anything you need in large quantities can be a real bargain on eBay compared to buying from a regular retailer, especially if you can find a Chinese warehouser selling it. The downsides of eBay are (a) that eBay's search engine is just miserable at finding generic parts like "100 ohm resistors", (b) only a very limited selection of electronic parts are available at all, (c) there's often no way to know the manufacturer or source of the parts, so quality can be unpredictable, and (d) the listings don't tend to give you the same level of detail you get on Mouser (e.g., the exact size of the part). Despite all of those drawbacks, I've had good luck with generic parts like MOSFETs and resistors - those can be much cheaper than buying the equivalent name-brand parts from Mouser.

Custom circuit boards

- OSH Park is a US PCB maker that specifically caters to hobbyists like us. They charge by square inch of board space - as of this writing, \$5/sq in, for three copies of the board, with shipping included. That makes them a fantastic bargain for small prototype boards. They also make it extremely easy to order, by letting you upload an EAGLE .brd (board layout) file directly (most of the other guys make you do some extra steps to generate special CAD/CAM formats). And they're in the US, so if you're also in the US, turnaround tends to be quick - you don't have to wait for international shipping or customs clearance. The only snag is that the per-square-inch pricing gets really expensive for larger boards.
- elecrow.com is a Chinese company that makes custom circuit boards in small batches (lots of 5 to 10 pieces) at bargain prices. Look under "Services" and "PCB Prototyping" in their category list. I've been using them for group orders of the Pinscape Expansion Boards, and all of the batches have turned out well. The downside, if you're not in China, is that the international shipping is expensive - more expensive than the manufacturing cost in most cases. But the boards are cheap enough that the overall price usually comes out to only about \$2 to \$3 per board when ordering minimum lot sizes. Like most PCB makers, Elecrow requires you to generate "Gerber" files (a special file format

for manufacturing use) rather than uploading the EAGLE design files directly.

That takes a little extra work and probably seems very intimidating if you haven't done it before, but it's not actually all that hard; the process is explained step-by-step in Chapter 93, Fabricating the Expansion Boards.

- PCB Shopper is a great comparison site for PCB manufacturers. The site lets you enter the details of your order, then provides quotes from a wide range of vendors.

3D printing

If you don't plan to buy a 3D printer at home, there are several excellent online 3D-print services that you can send your design out to.

- All3DP is a shopping service for 3D printing. Upload your design, and it'll give you price quotes from multiple vendors for different materials and process options, with direct ordering links. This has become the first place I check because of the excellent price comparison engine.
- Shapeways has been my top vendor for a long time because of their excellent materials and reasonable prices for small jobs. I've had several items made here with good results.
- 3D Hubs is another on-line 3D print service. In the past, they were the "Über for 3D printing" (connecting buyers with local sellers offering 3D printing services), but lately (2019) they seem to have dropped that model and switched to simply offering their own fabrication services.

The online services cost more than home printing in terms of materials, but of course that doesn't count the cost of buying the printer itself. Plus, the commercial vendors offer far superior materials to what you can use in a home printer. Home printers mostly use ABS and PLA, which are fine for prototyping, but not for functional parts, since they're brittle and tend to disintegrate if exposed to any friction. The commercial services offer nylon materials (such as PA12 and PA11) that are much more durable. Many also offer the newer MJF (multi-jet fusion) process, which seems to produce particularly tough and durable parts. I'd highly recommend considering MJF for any functional mechanical parts.

Custom laser cutting and CNC fabrication

- SendCutSend offers precision laser cutting for metals, plastics, and other materials, and CNC cutting for wood. They can also do custom bending of metal parts. They do a really great job and their prices are quite reasonable.
- Ponoko does custom laser cutting of a wide range of materials, including plastics and metals. This is a good option for custom flat plastic parts that require precision cutting. I've used Ponoko for several projects, including the acrylic face plate for my speaker/DMD panel, all with good results.
- TAP Plastics does laser cutting, and they can also do straight cuts with conventional equipment. The latter is a cheaper option for basic rectangular pieces like a translite cover or apron cover. TAP has numerous store locations on the west coast - if there's one in your area, you can avoid shipping costs by visiting in person.

Software source code

Many of the core software components in a virtual pinball machine are open source,

meaning that the source code is published for anyone to inspect, customize, and contribute to.

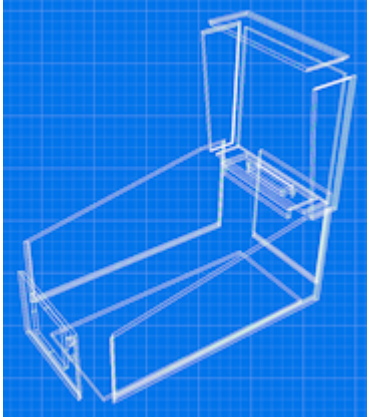
- Visual Pinball. The leading open-source pinball simulator and table design tool.
- DirectOutput Framework (DOF). System software that allows applications (Visual Pinball, PinballX) to control feedback devices in the cabinet (lights, solenoids, etc).
- B2S Backglass. Software that works with Visual Pinball to display animated backglass artwork, with the animations synchronized to the game play.
- Pinscape controller. KL25Z firmware for an all-in-one virtual pinball I/O controller, with plunger sensing, button input, accelerometer nudge sensing, and feedback device control.
- PinballY. A menu system and game launcher (also known as a "front end") for virtual pin cabs. This lets you browse your games, start games, and switch between games using a graphical arcade-style UI rather than the Windows desktop, to give your cab more of a finished arcade machine feel and disguise the fact that it's a Windows PC under the covers.

In case you're wondering about some obvious omissions in the list above, the following are **not** open-source: PinballX, HyperPin, Future Pinball, and the online DOF config tool. Those are "freeware", meaning there's no charge to use them, but their creators chose to keep the source code secret. That might not matter to you if you didn't want to see the source code, but I generally prefer using open-source programs even then, because of the greater assurance that the project can keep going if and when the original developer gets bored of it and stops working on it.

Pinball table information

- IPDB (the Internet Pinball Database) has detailed information on, and photos of, nearly all of the commercial pinball machines ever made.
- vpfforums has a collection of resources useful for creating new virtual pinball tables, such as playfield graphics, sound effect recordings, and 3D models of playfield parts. Click "Design Resources" in the top navigation bar for links to the various collections.

Part Two. *Planning and Building the Cabinet*



5. Road Map

Building a virtual pin cab is a big project. You shouldn't go into it thinking it's the light work of a couple of weekends. But it's also not an impossibly huge job, even though it can seem that way at times - especially during the research phase when you're trying to get your arms around all of the details.

Good organizers always say that the way to tackle a big job is to break it down into smaller pieces until the pieces are manageable. So let's look at the main sub-tasks that go into a pin cab build.

You don't have to attack these sub-tasks in the exact order listed here. Even so, we've tried to put things in a reasonable order that'll make the build process efficient. Some tasks are easier if others have been completed first, so you might find yourself backtracking or putting a job on hold if you tackle some of the later tasks on our list before completing some of the earlier ones.

Decide on the cabinet dimensions and TV size

You don't have to plan out your entire system in advance, but one thing you should settle early is the overall scale of your build and the size of TV you're going to use. Many other decisions depend upon the exact cabinet dimensions and the amount of space the TV will occupy, so you'll save yourself a lot of backtracking if you have hard numbers for these from the very beginning.

The thing that makes TV sizing tricky is that you can only buy TVs in certain sizes. Ideally, you'd be able to design and build your cabinet without giving too much thought to a TV, and then just drop in a suitable TV when the time comes. But when that time comes, you might find that all of the models you like are 1" too wide to fit, and the next size down is 4" narrower than you wanted.

This leads some cab builders to start by picking a TV, and then tailor the cabinet to fit the TV like a glove. But that's not for everyone. If you plan to re-use an old pinball cabinet or buy an off-the-shelf cabinet kit, you're stuck with standard dimensions. You might also want to use standard dimensions for the sake of faithful simulation, or simply because the cabinet hardware parts (lockdown bars, side rails) are cheapest and easiest to find in the standard sizes.

The same issues apply to a lesser extent to the backbox TV, although most cab builders aren't as concerned about an exact fit here since it's hardly the focal point of the game. My advice is to figure out the backbox TV separately after deciding on the main TV and cabinet dimensions.

We go into this in more detail in Chapter 7, Selecting a Playfield TV.

Go shopping

Once you know the basic scale of your system, you can start buying the main components. We provide a master part list for a fully decked-out cabinet in Chapter 20, Cabinet Parts List. You can choose a subset from that list that fits your own goals and budget.

Even with our master parts list to work from, you'll still have to do some original research, particularly for selecting the TV monitors and the PC components. There are too many options for both to allow simple one-size-fits-all recommendations, plus those product categories move so quickly that any concrete advice we could offer would be obsolete before you read this. Fortunately, apart from the TV and PC,

a lot of the rest of the cabinet can be built out of standard parts. The real pinball machine manufacturers were very cost-conscious, which drove them to build their games on common platforms with mostly interchangeable parts. The exterior shell of a virtual cab is almost identical to a real pinball (or can be, at least, depending on your design goals), so we can build our cabs out of those same interchangeable parts.

Build the PC

Now that you have some of the groundwork laid, you can start actually building something. Most cab builders like to start with the PC, probably because it's the most familiar territory to PC gamers. Plus it provides some (relatively) instant gratification, since it doesn't take too much work to get a new PC up and running.

If you've ever built your own desktop PC from scratch, you'll know exactly what's involved in building the PC inside your pin cab. The main work required is the research needed to pick out the parts: motherboard, CPU, graphics card, power supply, memory, hard disk. We'll tell you all the parts you need in Chapter 10, *Designing the PC*, and we'll offer some advice about how to select them, but you'll still have to do the research to select the exact components you want. Once you pick out the parts, it's pretty easy these days to do the actual assembly.

Set up the main PC software

The next step after building the PC is usually installing the core software, including the operating system and some pinball simulators. Most of the popular pinball software runs only on Windows, so that's the OS that almost all pin cabs use. We'll cover how to install the main pinball players in Chapter 15, *Pinball Software Setup*.

Build the cabinet body

Now we come to the cabinet itself, in the literal sense of the wood box that houses everything. This part of the build can be as simple as buying a new or used cabinet that's already assembled, or as elaborate as doing the woodworking from scratch. We'll cover the various options in Chapter 21, *Cabinet Body*.

Design and install the artwork

A pin cab is a significant piece of furniture to have in your house, so it's worth putting some effort into its exterior appearance. Some cab builders opt for a natural wood look to better fit in a domestic environment, and some choose a simple one-color paint job. For most of us, though, the aim is to replicate the look of a real pinball machine, which means using custom artwork in the distinctive graphic style of the real machines. One great way to get a thoroughly authentic look is with digitally printed decals. We'll cover the options in Chapter 22, *Cabinet Art*.

This is one of those steps where the order is fairly important. You'll want to get the artwork in place before you start installing the cabinet hardware or any of the insides of the machine.

Assemble the cabinet hardware

Once you have the wood shell of the cabinet built and finished with artwork, you can install the "hardware" - the side rails, lockdown bar, coin door, legs, and the parts

that attach the backbox to the main body. We'll go over the standard equipment and how to install it all in Chapter 23, Cabinet Hardware Installation.

Set up your power supplies

You'll need a standard PC power supply to power the motherboard and other PC components. If you're installing any feedback devices, you'll need additional power supplies for those. We'll explain what you need in terms of power supplies for the various components in .

Most cab builders also like to set up a power distribution system that turns power on and off across the whole system with a single button. We'll explain how to do this in Chapter 11, Power Switching.

This is all basic infrastructure that the rest of your system will depend upon, so it's a good idea to get this figured out and installed now, before installing any of the electronics. Another reason to do this early is that the power supplies take up a lot of space - it's good to reserve the space they'll need early so you don't find yourself having to move things around later to make room.

Install the PC in the cabinet

You can now finalize the PC installation inside the cab body. If you haven't already put together the computer components, this is a good time to finish that up.

Mounting the PC in the cabinet is usually straightforward. The main decision to make is what kind of enclosure you want to use: some people use a regular desktop case, but most cab builders use the cabinet itself as the "case", simply mounting the motherboard and other components to the cabinet floor or walls. We cover the possibilities in Chapter 10, Designing the PC.

Install the TVs

I think it's better to get the TV installed fairly early in the build process, but a lot of cab builders feel it needs to go in last, because it covers the whole top of the cabinet.

In either case, you should at least map out exactly where the TV will go before getting too much further, so that you can plan around its space requirements when installing everything else.

My recommendation is to install the TV in such a way that it can be easily removed at any time. If you do that, you can get everything in place for it early on, which will give you a very concrete idea of the space you need to carve out for it in the cab. But you can leave it out of the cab while installing the power supplies and feedback systems, so that you have easy access to the interior, knowing that you can pop the TV back in place when the time comes. And if you make the install/uninstall process easy enough, you can pop it in just to test clearances and fit from time to time.

We'll look at options for installing the playfield TV, including advice about how to maintain easy access to the cabinet, in Chapter 29, Playfield TV Mounting. That chapter includes a detailed plan for how to install the TV that achieves the goal of easy installation and removal, as well as allowing access to the cabinet interior for most jobs without even removing the TV.

Most cabs also have a second and possibly third TV in the backbox, for the backglass

and score/DMD display. We'll look at how to install those in Chapter 30, Backbox TV Mounting and Chapter 31, Speaker/DMD Panel.

Install buttons

You'll probably want to install and wire your cabinet buttons shortly after getting the cabinet assembled and the PC working, both for the sake of early play testing and because it's easier to do the installation work while the cabinet is still fairly empty. We'll go over which buttons you need and how to install them in Chapter 34, Cabinet Buttons.

Set up I/O controllers

I/O controllers are separate components - usually USB devices - that handle the connections between the PC and the unique devices in a virtual pin cab: buttons, plunger, accelerometer (for nudging), flashing lights, and mechanical and tactile feedback devices.

You don't have to set up all of the I/O controllers or functions at once. At this stage in the build, though, you'll at least want to set up the button input controller (also known as the key encoder), so that you can test out the newly installed cabinet buttons.

We'll look at the various controller functions and which devices you need in Chapter 12, I/O Controllers.

Install a plunger

The plunger can be installed at any point in the build, but you'll want to get it in place before you finalize the TV installation. The space in the plunger area is tight (even on a real machine), so it's worthwhile to do some measuring and planning. The plunger is close to the flipper buttons, and on a virtual cab it competes for space with the TV. We'll cover the details of installing the basic physical plunger as well as the options for connecting it to the software in Chapter 37, Plunger.

Install feedback devices

You'll probably find that you'll install the feedback devices in stages, rather than as all at once. Output controllers control devices individually, so you can easily set up a few now and add more later.

We'll cover the common types of devices, and how to set them up, in Chapter 44, Feedback Devices Overview.

Set up the sound system

The audio system in a pin cab is essentially just a PC desktop speaker system, but it has some special considerations. The main one is that most cab builders want to place the speaker drivers in the same places they go in the standard 1990s cabinet design. You can also embellish your system by using two independent audio systems - one for music and one for mechanical sound effects - or even a tactile subwoofer like the ones popular for home theaters and gaming chairs. We'll cover the options in Chapter 41, Audio Systems.

6. Serviceable Design

Real pinball machines have numerous design elements that are there purely for the sake of serviceability - that is, to make the machines easier to repair, upgrade, and maintain. These features are often the products of a series of refinements that wouldn't have been obvious without experience, and a lot of them are hidden, internal features that you wouldn't even realize are there when you're just playing the games.

It's not surprising that serviceability is such a high priority for the pinball manufacturers when you consider who their customer is. The thing to realize is that players are *not* their customer. Players are only the "end users". The *customer* is the commercial operator who buys the machines for their arcades and routes. The player mostly wants machines that are fun to play. An operator certainly cares about that, too, because an arcade game earns quarters by being fun to play. But what the operator cares even more about is high reliability and low repair costs. A machine that's down for repairs isn't earning quarters no matter how much fun it is.

Some of the things that they do in real pinball machines to make them serviceable translate readily into the world of virtual cabs, where we can copy them to get the same benefits. But new virtual cab builders are often unaware of how the real machines work, so they don't know there's a good design they can copy - instead they make things up as they go. And of course the first time you build *anything*, you're likely to come up with quick-and-dirty ways to solve immediate problems without considering the longer-term costs. So I want to point out a few of the serviceability features in the real machines that we can leverage for virtual use, so that you're aware of them, and so you can keep them in mind as you plan your build.

What makes a machine serviceable

Before we get to implementation details, let's look at what serviceability really means in the abstract. These are what I consider the key goals of a serviceable design:

- Accessibility. You should be able to access everything in the machine easily, without having to destroy anything, without having to de-construct anything, and ideally without any tools.
- Modularity. You should be able to remove and replace most individual parts or subsystems without having to destroy anything, and with as little work as possible. Electrical connections should be pluggable, for example, rather than being hard-wired or soldered; parts should be secured with removable/reusable fasteners like screws, rather than permanently glued or nailed.

Specific recommendations

With the abstract goals above in mind, here are some concrete recommendations for how to achieve them, based in part on how the real machines accomplish the same ends. Things are different in a virtual cab, naturally, but some of the ideas from the real machines carry over surprisingly well.

Liftable playfield

The real machines are set up so that the playfield can be tilted up from the front and lifted all the way up to lean against the backbox. This lets you access the entire interior of the main cabinet and the entire underside of the playfield *without taking anything apart*. It's like popping the hood of a car to get to the engine. It lets you

get in and out quickly. Minor jobs remain minor because it only takes a few seconds to open the machine up.

In my opinion, this is the ideal way to arrange a virtual cab as well. Accessing the interior of the main cabinet is at least as important in a virtual cab as a real machine, as that's where we typically install the PC motherboard and most of the feedback devices. In the virtual cab, it's the main TV (in place of the playfield) that needs to tilt up and out of the way like a car hood.

This is why I always advise against any design that involves the main TV being difficult or impossible to remove, such as installing the TV in routed grooves along the side walls.

Removable TVs

On the real machines, the playfield not only tilts up and out of the way, but can also be removed entirely. They even make this fairly easy: the playfield isn't actually permanently attached, but is only resting on the pivots that let it tilt up, so remove it is just a matter of lifting it off the pivots.

In a virtual machine, it's equally important to make the TVs removable. They are of necessity at the very front of every part of the machine, so you need to get past them to get to anything else. If it's hard to remove any of the TVs, it's hard to service what's behind it.

For the playfield TV, the ideal as far as I'm concerned is to install it analogously to a real playfield, with the same ability to tilt it up for smaller jobs and remove it entirely for larger jobs.

The backbox TV(s) should likewise be removable, and like the playfield TV, this should be possible without a lot of disassembly and certainly non-destructively (meaning you shouldn't have to rip off the sides or cut any new holes anywhere).

If you're using a separate DMD TV, that should of course be removable as well. This one is usually easy to make modular, at least, since it's so small.

Foldable backbox

On the real machines, the backbox is attached to the main body with a hinge that lets it fold forward, so that it lies flat on top of the main cabinet. This is a key feature to make it practical to transport the machine, since it's too tall, top-heavy, and fragile to move it with the backbox upright.

Some pin cab builders figure that they can just remove the backbox if they ever need to move the machine. Consider the amount of wiring that you'd have to disconnect to do that, and the risk of breaking something when redoing the wiring. If a folding backbox is impossible because of the geometry of your backbox TV, then at the very least, be sure that the wiring connectors are all modular, so that it's quick and reliable to reconnect the wiring.

Modular wiring connectors

Real pinball machines have lots of wiring internally, with many interconnections. (Stern Pinball has estimated that there's about half a mile of wire in a typical Stern machine from the 2000s. It might even be a bit higher in the 1990s machines, since they used lower-tech electronics to implement a similar feature complexity.) The real machines deal with their many connection points mostly by using plug-and-socket connectors.

In the 1990s machines, they made heavy use of a few different connector types

made by Molex. You see the term "Molex connector" used almost generically as though it referred to some specific physical plug type, but it's actually a particularly unhelpful term when you're searching for parts. Molex the company makes a huge array of diverse connector types, so "Molex connector" can refer to all sorts of things. If you want to be helpful when describing a particular connector to someone, you need to give them a Molex part number or at least the Molex product line name.

Pluggable connectors of the sort used in the real machines have two important virtues:

- You can easily connect and disconnect them at any time
- You can't get a connection wrong when re-connecting something, because the plug only fits in the one place where it's supposed to go

Most virtual pin cab builders intuitively recognize the importance of modular wiring - of using some kind of removable connectors rather than soldering everything together permanently. But many new pin cab builders gravitate towards screw-terminal blocks, and in my opinion, those don't quite achieve the full goal here. Terminal blocks do avoid the need for permanently soldered connections, so they address the first point above right, but they miss the mark on the second point. Consider what happens if you have a couple of terminal blocks for wiring a particular part, and you disconnect the wiring: where do the wires go when it's time to reconnect them?

Proper pluggable connectors of the sort we're talking about are definitely a bit more work to set up than screw terminals. But they're great once they're in place, because you can arrange things so it's practically impossible to plug things in the wrong way. These connectors can also be time-consuming to select during the design phase, because there are so many options available. I'm convinced you could build a nice engineering career on a solid knowledge of the Molex catalog and of which connector to use when. I've tried to make your job here a little easier via pointers to some good basic options in Chapter 80, Connectors.

There are three clever techniques used in the real machines to make the connections "idiot-proof", which are worth cribbing when you're wiring your virtual cab:

- Whenever possible, use a unique connector type. It's impossible to plug a nine-prong Molex .062" plug into an eight-prong socket. If there's only one nine-prong Molex .062" plug and one nine-prong socket in the whole build, that's one connection that you can't plug back in to the wrong place.
- Wherever you have to use the same connector type more than once, use a "keyed" connector. That means that you snip off one pin on the plug side, and block the corresponding socket on the receptacle side. If you try to plug the wrong 9-pin plug into the wrong 9-pin receptacle, that wrong plug won't have the right pin clipped, so it won't fit into the blocked socket.
- Use plug-and-socket connectors that only plug in one way, so that you can't accidentally plug something in backwards. You get this benefit automatically for connector types that have asymmetrical shapes. All of the Molex .062 and .093 connectors are inherently designed this way, for example.

7. Selecting a Playfield TV

For most cab builders, the playfield TV is the most important piece of hardware in the system. A pin cab is, after all, fundamentally a video game. The weight of this importance makes it tough to decide on the perfect TV, but it's even more complicated because of the physical constraints of a pin cab, and the special performance demands of video gaming.

I'd love to simplify this by offering a list of Amazon "buy it now" links for the best TVs for the job. Unfortunately, I really can't, because any such list would be out of date by the time you're reading this. TV product life cycles are only about six months these days.

In fact, even "real time" recommendations on the forums can go stale quickly. If you talk to someone who already finished their project, they probably bought their TV at least a few months ago, so their particular model might already be hard to find. Your best bet is usually to share notes with other people who are out shopping right now.

Since I can't give you a list of models to choose from, I'll instead try to offer some advice to help you figure out what to buy. This section attempts to answer the questions that new pin cab builders often have when looking for a TV (and maybe answer some questions you didn't know you should ask).

Note that this chapter is about *selecting* the playfield TV. We'll get into the details of actually installing it later, in Chapter 29, Playfield TV Mounting.

Size constraints

Whatever else you look for in a TV, it has to fit your cabinet. Obviously that means it can't be bigger than the available space. Most people want a TV that's as big as possible within that constraint, to minimize any "dead space" not covered by the TV image.

There are two ways to approach this problem of finding the ideal fit:

- Pick a TV that fits your cabinet
- Build a cabinet that fits your TV

Most people go with the first approach, because they've already decided to use the standard dimensions of a real pinball machine. Using standard dimensions is important if you want it to look authentic, since the real machines all come in about the same size and have recognizable proportions. Building to a standard size also lets you use off-the-shelf pinball parts for your cabinet hardware (lockdown bar, side rails, legs, and so on), which is another key part of making it look authentic.

Not everyone feels compelled to use standard dimensions, though. If you're doing your own woodworking, you can tailor the machine's dimensions for a custom fit to any TV of your choosing. This gives you more flexibility in picking out a TV. The tradeoff is that a non-standard size and proportions can harm the illusion that it's a real machine. You'll also need to buy custom parts for some of the cabinet hardware, since off-the-shelf parts are sized to fit the standard cabinet widths and lengths. Custom parts are almost always more expensive than standard parts and can be harder to find.

In either case, whether you're picking a TV to fit your cabinet or sizing your cabinet to fit a TV, the dimensions that matter are the **inside width** of the cabinet and the **exterior height** of the TV. You're going to turn the TV sideways to mimic the layout

of a pinball playfield, so the height of the TV has to fit across the width of the cabinet.

Note that the **width** of the TV isn't a constraint in most cases. Normal pinball playfields are considerably more elongated than 16:9 TVs, so any TV that fits into the available cabinet width will easily fit front-to-back, with room to spare.



Leftover front-to-back space

As mentioned above, a 16:9 TV will fit front-to-back in a normally proportioned cab with room to spare, meaning there will be some extra space that the TV doesn't fill. Assuming you find a TV that's nearly as big as possible for the cabinet width, the extra space will amount to about 6" in a standard body cabinet, and about 7" in a widebody.

Some people are bothered by that leftover space, and some aren't.

Before you decide that it's a problem, consider that you can put the space to good use. If you're using a plunger, it will jut into the front of the cabinet by about 3", which might necessitate moving the TV back that far. You can fill the gap that creates with an "apron", similar to on a real pinball machine, with an instruction card and price sheet. A 3" apron at the front still leaves 3" to 4" at the back. This is an ideal place to put a row of flasher domes, for bright lighting effects during play.

Some new cab builders get very fixated on the idea of covering every available millimeter with video display, and especially hate the idea of any extra space at the front. They insist on having the TV start exactly zero millimeters from the lockbar. I understand this instinct; I had the same thoughts myself when I was first building my cab and discovered the space conflict between plunger and TV. I ultimately decided that the plunger was important enough to justify the space at the front. Once I had everything assembled, I found that it makes absolutely no difference when playing to have the TV set back a few inches. If you think about it, you'll see why: your brain pays attention to the parts of the visual field where the action is taking place, and essentially makes you blind to the rest. It's the same effect as

when you're watching a movie: you really don't see the curtains around the screen, you just see what's on the screen. A few inches of "curtains" in the form of an apron won't even register visually during play.

If you're still absolutely certain you can't live with any leftover space front-to-back, there are a couple of options for eliminating it:

- Build your cab to a non-standard size, shortening it from the standard length to eliminate the excess space. This can make the proportions of the finished product look unusual or "wrong" if you're used to seeing the real machines, but that doesn't bother everyone, and some cab builders prefer that to the excess space. A non-standard length also means you can't use a standard pinball glass cover or side rails.
- Use an ultra-wide TV instead of a 16:9 set. A few TVs are available with 21:9 aspect ratios. That's actually even more oblong than standard playfields (which are about 18.7:9), so it goes too far the other direction, but you could tuck some of the extra TV length into the area under the backbox.

This approach has some downsides. For one, it's hard to find ultra-wide TVs in our size range. The format never caught on with consumers, so there aren't very many models available. For another, you might be making things hard on yourself when it comes time to setting up software. Almost everyone uses 16:9 TVs or monitors for playing pinball, so most of the software assumes that layout.

Picking a TV based on cabinet size

If you're basing your design on a pre-determined cabinet size, you need to pick a TV that fits the cabinet.

TV sizes are always stated in terms of the "diagonal" size, which is the distance between diagonal corners on the display area. However, recall that the relevant dimension for fitting to a pin cab is the TV's **height**. How do you translate between height and diagonal size? You can get a rough approximation using this formula:

$$D = 2.04 \times (W - 1)$$

W is the inside width of the cabinet in inches (the distance between the inside surfaces of the cabinet side walls), and **D** is the nominal diagonal size in inches of the biggest TV that will fit. This formula assumes a ½" bezel all around.

But that's only an approximation, because manufacturers always round the diagonal size up, and because the size of the bezel varies from model to model. So use the formula as a guideline, not as an exact specification. Shop for TVs with a stated diagonal size within an inch (plus or minus) of the size you get out of the formula. Then check each TV's specifications to get its actual height.

When you're looking at the TV specs, the one to pay attention to is usually called "height without stand". Most flat-screen TVs come with a stand that you can choose to attach or not. In our case, we won't need it, since we're going to lay the TV on its back rather than stand it up on a tabletop.

Applying the formula, we get the following results for the standard cab dimensions:

Type	Cab inside width	Max. TV size (diagonal)
Standard body	20.5"	39.8"

Wide body	23.25"	45.4"
-----------	--------	-------

So if you're building a standard body cab, you should be able to fit most 39" TVs, and possibly a 40" TV, if it has a narrow enough bezel. For a widebody, you can fit about a 45" TV.

Building a cab around the TV

If you're willing to customize your cab dimensions to fit your TV, you're much less constrained in your TV options. You can go out and find the perfect TV first, then measure it and design your cabinet plans around the TV's dimensions.

I'd recommend adding ¼" to ½" to the exterior height you measure for the TV to get the cabinet inside width. This will give you a little extra room for getting the TV in and out of the cabinet.

Remember that the TV height determines the **inside** width of the cabinet, but most other dependencies are tied to the **outside** width. The width of your lockdown bar, front and back cabinet walls, and glass cover all depend on the outside width. If you're using standard ¾" plywood, the outside width will be 1.5" wider than the inside width.

Custom-width cabinet hardware

There are two main pieces of cabinet hardware that depend on the cabinet width: the lockdown bar and the glass cover.

You can buy a custom-made lockdown bar with a tailored width from VirtuaPin and others. Search for "custom lockdown bar". The prices on these are about twice the price of the standard lock bars, but it will let you create an authentic look for your custom cabinet.

You won't be able to find custom-width playfield glass from pinball vendors, but it should be easy to find locally from any window glass shop. Ask for 3/16" tempered glass. Window glass vendors should be able to cut this to any custom size for you. Alternatively, you can use acrylic (plexiglass), which you can buy in custom sizes from local vendors like TAP Plastics.

Squeezing in a too-big TV

The perennial question that new cab builders ask is: how do I cram in a TV that's slightly too big for my cabinet design?

Part of the reason this comes up so often is that you can't buy a TV in just any size. You can only buy a size they actually sell. It's unlikely that you'll find a TV for sale that's a perfect fit to any standard cabinet plans. So you have two options: (a) you can pick a TV that's slightly smaller than the ideal, which (being smaller) will easily fit, but which (being smaller) will leave an unsightly gap around the edges. Or (b) you can pick a TV that's slightly bigger than ideal, and find some "hack" to make it fit.

The other part of why this comes up so often is that most new cab builders hate option (a) and believe they won't be happy unless they find a way to cram in a too-big TV.

My advice is to suppress your knee-jerk reaction to option (a). When we were

considering the related problem of the leftover front-to-back space earlier, I mentioned that you won't really notice the space while playing, because your brain tends to focus so much on the action and ignore the periphery. Well, the same thing applies to leftover space side-to-side. Despite what your instincts might tell you, it really won't make much of a difference during play if you leave a little blank space on each side of the screen. In fact, if you look closely at real pinball machines, you'll observe that they give up about half an inch on each side of the playfield for wood rails around the perimeter.

What you gain by going with the "next notch down" option is an easy fit, a simpler design, and the ability to maintain easy access to the cabinet interior after the TV is installed. I consider these to be important features.

Okay, I tried. I know some people just can't be convinced of this. So what if you have your heart set on a TV that's a little too big? Is there any way you can squeeze it in without redesigning the whole cab? Yes, there are some options.

De-case it

One approach is to "de-case" the TV - remove the outer plastic case and just use the internal LCD panel.

A few years ago, this was practically a standard practice among cab builders. At the time, the plastic cases were quite a lot larger than the panels inside, so the only way to get a reasonable fit was to take the cases off.

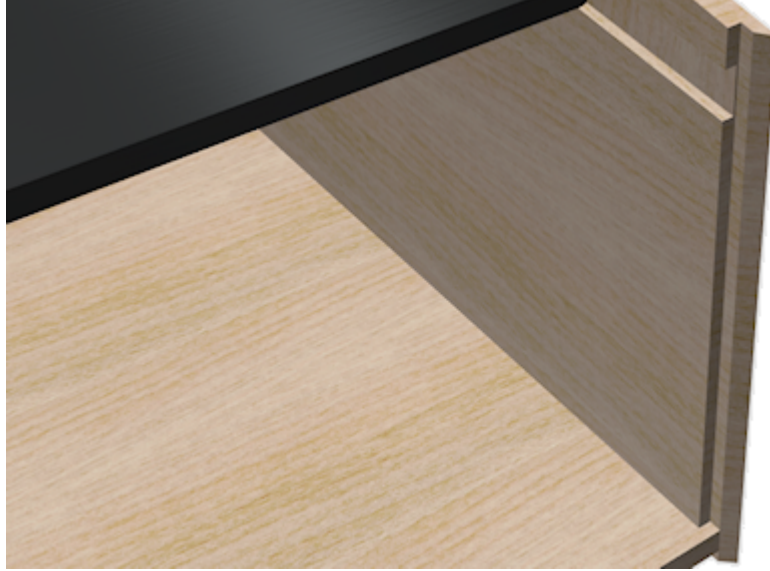
Times have changed, though, and most cab builders now leave their TVs intact. There are two main reasons for this. The first is that cases have shrunk to the point where they're practically no bigger than the panels inside, so de-casing doesn't offer a meaningful size reduction. The other is that many newer TVs simply can't be de-cased without damage. The way manufacturers have managed to make modern cases so svelte is that they've removed the internal supports that made older models bulkier. That means the cases themselves now have to serve as exoskeletons that hold everything together. There's a big risk of cracking the delicate glass panel that holds the LCD elements if you remove the structural support provided by the case.

I'd advise against de-casing for any newer set. If you want to attempt it despite the risk, I'd try to get advice first from someone who's disassembled the same model. The pin cab community is small enough that you probably won't find anyone there, so you might try casting a wider net. For example, perhaps look for someone who's successfully repaired the type of TV you have.

Route grooves for the TV

Another way to make a slightly-too-big TV fit is to make the cabinet a little wider on the inside, but only where the TV goes, by routing out grooves in the side walls wide enough for the TV. Here's how this might look:





I'm not a big fan of this approach for two reasons. First, it weakens the side walls. Second, it makes it much more difficult to remove the TV if you want to access the inside of the cabinet for repairs or upgrades. I see easy access to the interior as a top priority. If you use routed grooves, you'd have to remove either the front wall or the back wall of the cabinet to take out the TV, and to do that you'd have to take off the legs. That's enough to make me rule out this approach if it were my own cab.

A similar alternative is to route out grooves like this all the way to the top of the side walls. That would at least let you remove the TV from the top, but it would weaken the walls even more than simple grooves.

Despite my strong reservations, routed grooves like this are fairly popular among cab builders. But the tradeoffs are too onerous for me to recommend this approach.

Use thinner plywood

Rather than routing grooves, you could simply use thinner plywood for the walls. That would increase the inside width without changing the exterior dimensions. You'd still be able to use off-the-shelf hardware (like the lockbar), since that's all sized according to the exterior width.

One downside of this approach is that the cabinet would obviously be a little less sturdy. But that's probably okay for home use, since your cab won't have to stand up to the punishment a public arcade machine receives. The other downside, probably more important, is that flipper buttons and some other parts are sized for the plywood width, so you'll have some ill-fitting parts to deal with.

Also, keep in mind that you'll have to make adjustments to the carpentry if your plans call for miter joints or the like. Joint dimensions will depend on the plywood width.

TV features and performance

So far, we've been focused exclusively on picking the right size of TV. But that's hardly the only criterion. You also want a TV that displays a good image, and one that works well for games, which have somewhat different characteristics from ordinary video sources.

Let's look at some of the specific features to consider, and the performance metrics you should pay attention to.

1080p vs 4K vs 8K

1080p HD TVs were the standard for pin cab playfields for a long time, largely because that was the highest resolution we could get in this size range. Starting around 2017, though, the industry started moving towards the "Ultra HD" standard, also known as "4K". And in mid 2019, an even newer generation known as "8K" has started to become available.

The difference between 1080p, 4K, and 8K is pixel resolution. In other words, the pixels on 4K sets are smaller than on 1080p sets, and the pixels on 8K sets are smaller still. A 4K set has four times the number of pixels per unit area as 1080p, and 8K has four times the pixels per unit area as 4K. The smaller the pixels are, the harder it is for the eye to discern individual pixels; smaller pixels blend together better to make a more realistic image.

Higher pixel resolution comes at a cost in performance, though (in addition to the higher dollar cost). More pixels means more work for the PC. The PC has to fill in every pixel on the display on every video frame, so the larger number of pixels means the PC has to do more computation on every frame. If you use a 4K TV, you'll need a more powerful CPU and graphics card to keep up with the higher computational load. So if you want to use 4K, you'll need a more powerful and thus more expensive computer rig. 8K likewise requires a more powerful computer than 4K.

Recommendations

If I were building a new cab right now, I'd go with 4K for the playfield TV. It's well supported by the operating system and pinball software, and the price premium for a 4K TV over a 1080p TV isn't that large. You will have to spend more for a 4K-capable video card, but even that is entering the mainstream, and enough options are available that the price doesn't have to go into the stratosphere.

I wouldn't go as far as 8K right now, though. It's much more expensive than 4K right now, and I'm skeptical that it will even make much of a visible difference in a pin cab application, since at this viewing distance, 4K is already approaching the limits of the human retina's ability to resolve pixels. (Although I'm sure some people will be able to see the difference.)

Finally, on the off chance you come across a 720p set, skip it. 720p used to be common in this size range; it's almost extinct at these sizes now, but you might still see a few on sale. They're cheap, but they're really not suitable for the playfield. 720p simply isn't adequate resolution for the viewing distance of a playfield TV. (720p *is* generally just fine for a backglass TV, though. That's a smaller TV at a greater distance, and the graphics it displays aren't as demanding.)

LCD, LED, QLED, OLED

There are currently two main display technologies available: LCD and OLED. There's also an older flat-panel technology called plasma that's not currently being manufactured, but you might still see used sets or remainders available. Here's a brief overview of each panel type.

LCD: This is currently the most common display type. An LCD panel uses liquid crystal pixels that can range between (almost) opaque and (almost) transparent. A backlight is placed behind these pixels. When the liquid crystal turns opaque, it looks black (or at least dark gray) because it's blocking the light from the backlight. When it turns transparent, it looks white because it lets (most of) the light from backlight through.

LED: This is really the same thing as an LCD TV, but it uses an LED-based backlight

instead of the fluorescent backlights used on older LCD TVs. "LED" is a marketing term that the manufacturers use as an intentional bit of misdirection, because they know that consumers think of LCDs as an older, boring technology. But an LED TV actually is an LCD TV by a different name.

QLED: This is yet another marketing term for an LCD TV. In this case, it's an LCD panel with a special type of LED backlight called a QLED or quantum-dot LED. Quantum sounds even more cutting-edge than LED, doesn't it?

All of these LCD TV types, whether the manufacturers call them LCD, LED, or QLED, are fundamentally the same backlight-and-shutter design. The fundamental weakness shared by all LCD panels is that the shutters can't turn 100% opaque, so they can't display true blacks, just varying shades of dark gray. Some panels are better at this than others, and it's one of the big quality differentiators among LCD models. LCD panels also have inherent limits on viewing angle because of the way light has to be funneled through the shutters. Again, some models are better at this than others.

The backlight type does make some difference. LED backlights generally produce better color fidelity than fluorescent tubes did, and they use less power and run cooler. All of that is great for a pin cab, so if you're considering an LCD TV, I'd definitely give priority to the LED models. But you'll hardly have to even think about that since practically all of the TVs in this size use LED backlights. QLED backlights supposedly have even better color fidelity than regular LEDs, according to the manufacturer's claims, but I haven't seen any independent testing confirming this.

OLED: This is a truly is a different display type, not just a variation on the LCD. An OLED panel is an array of small "organic LED" pixels, each of which can be turned on or off individually. There's no backlight, since the OLED pixels emit their own light directly. ("Organic" doesn't mean that they grow them without antibiotics and pesticides, but rather refers to the chemical components making up the emitter.)

On paper, OLED has big advantages over LCD. Producing light at the pixels rather than blocking light with a shutter allows for true blacks, which makes for higher contrast and better-looking images. Emitting light directly at the display surface (rather than blocking light from a backlight) allows for unlimited viewing angle. However, OLED is still a relatively immature technology, and reviews of current models are mixed. There are several potential drawbacks. The first is brightness: current OLED models are only about half as bright as LED-backlit LCDs. The second is display lag. Console gamers have reported substantial lag in many available OLED sets. A third is "burn in", where pixels get "stuck" if a static picture (like a pinball playfield!) stays on the screen for too long at a time. Early OLED models also had problems with pixel lifetime, which was particularly problematic in that the color components in the pixel can degrade over time at different rates, causing the color balance to change as the panel ages. Newer OLED panels will probably have better longevity and color stability, but I'm not sure the problem has been completely solved yet. In any case, don't dismiss OLED because of these concerns. These are just things you should dig into when you're researching models. These concerns might disappear entirely over the next few model years as the technology matures.

Plasma: There used to be yet another display technology known as plasma. These used gases trapped in tiny glass cells to generate light. As in an OLED, the individual pixels emitted light (rather than blocking light like in an LCD), so plasmas had many of the same virtues as OLEDs. But they were never as popular with consumers as LCDs, and never as cheap to manufacture, so the electronics companies eventually all stopped making them (the last ones were built around 2015). Plasmas generally had excellent picture quality, but they had a couple of drawbacks for virtual pin cab use. For one, they generated a lot of heat; for another, their glass panels were fragile

and not meant to support their own weight when laid on their backs, as we need to do in a pin cab. I'd avoid them for pin cabs as a result. But it's really moot now given that you can't buy them anyway.

Recommendations: Most of your options in our size range will be LED-backlit LCD TVs. Fortunately, that also happens to be an excellent choice for our needs. It's a mature technology that the TV manufacturers have gotten very good at building, so many excellent TVs in our size range are available.

I'd also consider OLED if you can find a suitable model. I think OLED will eventually be a superior option, because the light-emitting pixels are inherently superior to the shutter-based LCD design for producing high contrast and for wide viewing angle. However, there aren't many OLED models available yet, so your options will be limited. They're also more expensive, and the technology might not be mature enough yet to be an ideal fit for gaming. Be sure to look carefully at the concerns mentioned above relating to OLED, particularly display lag and image retention. If you find an OLED you like, do some research on the Web to see if any console gamers have experience with it, since console gaming places the same demands on a TV as virtual pinball.

Flat vs. curved screens

It almost goes without saying, but a pinball playfield is best simulated with a flat-screen display.

This is generally an easy requirement to fill with current TVs, since most LCD and OLED models have perfectly flat screens. But some models are now available with a convex curvature across the width of the panel. This is supposed to give you a wrap-around effect like in a large-format movie theater. Some people like the effect, others see it as little more than a sales gimmick. Whatever your feelings about it for a living room TV, though, I'd recommend against it for a virtual pinball playfield TV. A playfield TV is oriented in portrait mode, which defeats the purpose any wrap-around effect. The curvature will only serve to distort the geometry of the image.

Input lag

One of the really important differences between video gaming and regular TV viewing is that gaming is interactive. The animation on the screen responds to actions you take in the game. This exposes an element of TV performance that's not noticeable in normal passive viewing: "input lag". This is the amount of time that passes between the TV receiving the electronic signal for a video frame, and the video frame actually appearing optically on the display panel.

Input lag is important (in a bad way) to video gaming because it creates a time gap between when you press a button and when the resulting action appears on screen. If the time gap is long enough for you to perceive, it makes the gaming action feel leaden and unresponsive. You want the flipper to flip the instant you press the button, not a couple of seconds later after the ball has already rolled off the end!

Don't confuse input lag with "refresh rate", "response time", or "pixel cycle time". The refresh rate refers to how many times per second the TV draws a video frame. The response time or pixel cycle time refers to how quickly a physical pixel can change color. These times are important in their own right, because they affect how smooth motion looks on the display. But they're entirely different things unrelated to input lag.

Where to find input lag numbers

I've never seen a manufacturer list input lag in their spec sheets, so you have to dig

a bit to find information on it. Manufacturers do often quote pixel cycle times, response times, and/or refresh times, but remember that input lag isn't in any way related to those.

Your best bet for finding concrete data on input lag is console gaming Web sites, since console gamers use regular TVs like we do. One good site is displaylag.com. They measure input lag with special equipment and post the numbers on their site. They have a large database of current models that they update regularly.

What's an acceptable input lag?

Short answer: 40ms or less.

You don't need a TV with zero input lag, and it's impossible to find such a thing anyway. As long as the actual lag time is below a certain threshold, you won't be able to perceive any lag time at all, so anything below that threshold might as well be zero.

Human time perception varies according to context, but for video gaming, the main thing that matters is action/reaction timing. An action/reaction sequence is something like this: You push a button. A light appears on screen. Did the light appear exactly when you pushed the button, slightly before, or slightly after? When researchers do this experiment, they find that time gaps of up to about 50ms are perceived as exactly simultaneous. In other words, humans can't tell the difference between truly simultaneous and about a 50ms delay. It's not a matter of how smart you are or how closely you're paying attention; it's simply a fact of human nervous system physiology. Our neurons can only move signals so quickly, and as a result our brains perceive events that are very close together in time as though they were perfectly simultaneous.

This doesn't mean a TV with a 50ms input lag time is automatically good enough. You don't perceive the TV's lag time in isolation, but rather in combination with all of the other sources of latency in the overall system: delays from the key encoder device, the USB connection, the Windows video drivers, the pinball software itself. The latency from these other components varies, but in a well-tuned system it might add up to around 10 to 20ms. So that leaves us with 30 to 40ms to work with for the TV.

What causes input lag?

Input lag is caused by the internal digital processing that the TV does to the image before realizing it on the display. Most of this is processing that enhances the picture in some way: resolution up-scaling, frame interpolation, sharpness enhancement, noise reduction, motion smoothing. Modern TVs all do these enhancements digitally, by putting the pixels into a memory buffer inside the TV and running some software algorithms over the pixels. The software processing takes time, just like on a PC, and that processing time is what causes the lag.

Note that input lag has nothing to do with the physical pixels, so you can't guess anything about input lag based on what type of panel technology the TV uses. LCD, LED, OLED, plasma - none of those are inherently faster or slower in terms of input lag. It's purely a function of the digital image processing going on inside the TV.

How can you minimize input lag?

The best way to minimize input lag is to buy a TV with low input lag. You can't generally find this information on manufacturer spec sheets, but you can check gamer Web sites like displaylag.com. As described above, you don't need a TV with zero input lag (such a thing doesn't exist), you just need a TV with input lag low enough to be imperceptible. I'd use a threshold of 30ms to 40ms, and rule out sets

with much higher lag times.

Definitely stay away from sets with unusually high lag times. Some TVs currently on the market have lag times above 100ms, which will be maddeningly obvious during game play.

Even if your TV has great lag time numbers on paper, you'll still need to adjust its menu settings to get the best performance out of it. Even the fastest TVs can have bad lag times when all of their picture enhancement modes are enabled, and all of those modes are usually enabled by default when you first take your new TV out of the box. Every TV has its own menu settings that affect lag in different ways, so you might need to do a little Web research or experimentation, but here are a couple of rules of thumb applicable to most TVs:

- Turn on "Game Mode". Most TVs have a few master modes you can select from, with names like Movie, Pro, Vivid. One of these is usually a Game mode. If your set has such a mode, select it. In most cases, the main purpose of this mode is to minimize input lag, so it's the easiest way to go straight to the right settings on most sets.
- Turn off all picture and motion enhancement features: sharpness, noise reduction, high frame rates (120Hz or 240Hz, for example), and especially anything related to motion smoothing. Motion smoothing is the worst offender because it usually involves buffering up several frames for interpolation purposes, which deliberately delays the display by that many frames.

Effect of connector types on lag

In some cases, you might see different lag times with different connector types. Most newer TVs use HDMI connectors exclusively, so you might not have any other options. But if your TV has a mix of connector types (HDMI, DVI-D, DP), and you can't eliminate lag via mode settings, you might try different connector types to see if one type is better than the others.

There's nothing inherently good or bad about any of the connector types that affects input lag, so don't look for a rule like "DVI-D is fastest". Any such claims you see on the Web would only apply to a particular TV model, if they're even true. The only reason connectors would have any effect is that the internal electronics in some TVs have a faster path for some connectors than others.

Picture quality

This is probably the easiest metric to find opinions on, since everyone buying a TV for any use cares about it. You can simply look at user reviews on Web stores that sell the TV to get an idea of what people think of different models. For professional reviews, you can check Web sites and magazines that specialize in consumer electronics.

Basic video picture quality is generally excellent for most newer TVs, so user reviews are more useful for ruling out the occasional problem model than for distinguishing among the best models.

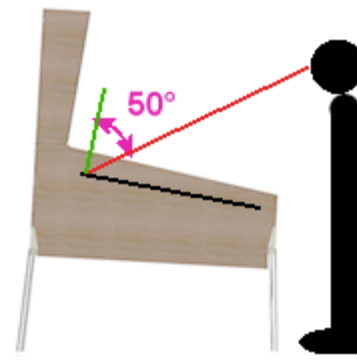
Viewing angle

Some types of displays produce a better image when viewed head-on than when viewed at an angle. LCD panels tend to have this property. Viewing from a steep angle can make the picture look dimmer, washed out, or uneven.

The position of the playfield TV in a full-sized cabinet creates an off-axis viewing angle of about 50° to 60°, depending on the height of the player, so it's important to

find a TV that maintains its image quality when viewed from that kind of angle.

Unfortunately, the manufacturer claims for viewing angles in the specifications aren't usually helpful, because they only tell you the range where you can see any image at all. In fact, they usually quote the viewing angle as 180° , which is just the maximum for viewing a planar surface. We're really interested in the range of angles where the picture quality holds up without significant loss of brightness or uniformity. The best way to check is to look at the set in person and specifically try viewing it from about 60° off axis.



If you can't check your candidate models in person, you can at least check user reviews for any red flags about viewing angle. Viewing angles are generally excellent in newer 1080p and 4K LCD panels, and people have come to expect this, so other buyers will probably have noticed if a model has any problems with this.

Note that viewing angle is almost never an issue with OLED or plasma displays. These technologies have their light emitters located directly at the surface of the display, which makes them viewable from any angle.

Motion artifacts

Some TVs are better than others at displaying moving objects realistically. Pinball simulation obviously involves a bunch of rapidly moving objects, so motion rendering is an important element of the overall picture quality in a pin cab TV.

When a TV doesn't handle motion well, you'll perceive effects known as motion artifacts:

- Blur (a moving object looks fuzzy)
- Ghosting (a moving object looks washed out or partially transparent)
- Jitter or judder (objects jerk or vibrate rather than moving smoothly)

It's commonly understood that the "pixel refresh time", also known as "response time", tells you how well a TV renders motion. Yes and no; the refresh rate is important, but it doesn't tell the whole story. Don't get too attached to the idea that you can just look for a TV with the fastest pixel update speed and call it a day. One problem is that there's no standard way to measure these values, so manufacturers can pick whatever measurement is the most favorable; this makes it fairly meaningless to compare the numbers for different models. The other issue is that the apparent smoothness of motion depends on other factors besides the pixel response time. It's more complex than that because motion perception happens in the human visual system, not in the TV. Motion artifacts like those listed above are caused by the interactions between your visual system and the display technology. Faster refresh rates generally reduce these artifacts, but other factors contribute to the artifacts as well, so refresh rate isn't a perfect proxy for motion rendering quality.

The best way to determine a TV's motion handling is (as always) to view it in person with suitable content. If possible, watch the TV in action playing a pinball video game, or some other video game with small moving objects against a fixed background. If that's not possible, try ESPN - sports tend to have a lot of motion of the right sort.

If you can't check the TV in person and you can't find another pin cab builder using the same TV, try user reviews on Web stores. Motion rendering is important to

regular TV viewers, especially sports fans, so you should at least be able to check for complaints about particular motion artifacts or problems.

Image retention

Some TVs suffer from a problem known as image retention, or "pixel burn-in", where pixels get "stuck" if you leave a static image on the screen for too long. This leaves a sort of ghost image stuck on the screen. This was a major problem in the ancient days of CRTs. This is, in fact, why they invented "screen saver" programs. The job of the screen saver is to keep varying the image displayed so that no one pixel will ever be held on at the same color for long periods.

Image retention has always been a concern for gamers because many video games have portions of the image that are fairly static for long periods. For example, console games often have score displays and on-screen controls that are always in the same place. Pinball is even worse in that most of the playfield just sits there motionless most of the time.

Fortunately, image retention is practically non-existent for LCD panels. If you're considering an LCD TV (whatever the backlight type - LED, QLED, fluorescent), you'll probably be immune from any concerns about pixel burn-in.

OLED sets are a different matter. Some OLED TVs are reportedly affected by image retention. If you're looking at an OLED model, look for reviews from console gamers to see if anyone has had problems with image retention on that model.

8. Selecting a Backbox TV

Most virtual cabs use a TV to simulate the backglass artwork of the real pinball machines. The backglass art is a distinctive and universal feature of pinball, and an important part of the aesthetic, so it's a must for most cab builders to replicate it in our virtual systems. It also serves a practical purpose, in that it's where many games display the score.

This chapter will try to help you design your backbox layout and pick a TV for it. We're just in the planning stages here; we'll get into the details of actually installing everything in Chapter 30, Backbox TV Mounting.

Choosing the right backbox TV

If you've just gone through the Chapter 7, Selecting a Playfield TV chapter, you're probably exhausted from thinking about all of the complex technical criteria that go into picking the right TV for your main cabinet. Fortunately, the backbox TV is a lot less demanding in terms of tech specs.

Really, the only important factor in choosing a backbox TV is size. You have to pick a TV that will fit in your backbox and fill the space it's supposed to cover. Most of the rest of this chapter is devoted to helping you decide what type of backbox layout you'd prefer, and to helping you determine the right TV size for your selected layout.

You can mostly ignore the rest of the technical factors that are so important to the playfield TV. The backglass area isn't part of the physical action in a real pinball machine, with the exception of a handful of tables with extremely novel designs, and even for those it's only a very small part of the action. For most games, it's mostly decorative, and shows mostly static images. So it's not all that important to find a TV with fast motion rendering or low input lag.

You don't even need a very high res monitor. Most people find that a 720p TV is perfectly fine for the backbox TV. The backglass artwork from real pinballs is mostly hand-painted graphics, and that kind of source material tends to look good even on lower resolution displays. The main picture quality features I'd look for are good black levels and color accuracy; those are more important than pixel resolution for cartoon graphics. Also consider viewing angle, so that the image doesn't fade too much when you're standing off to the side.

Virtual backglass options

There are two very different ways that you can set up your cab's backglass TV. Before you start picking out a TV, you should decide on one of these configurations, since it will determine the TV size you need.

The two options are commonly known as the **two-monitor** and **three-monitor** configurations.

The **three-monitor setup** mimics the physical layout of real pinballs made from about the mid 1980s. That's when the real machines started using a "speaker panel", a separate section at the bottom of the backbox containing the score display and speakers. Nearly all pinballs made after about 1984 used this arrangement.

For virtual pinball, this is called the "three-monitor" setup because it means you'll have a total of three video displays in your cabinet: the main one for the playfield, the TV in the backglass area, and a small monitor in the "DMD" (dot matrix display) area. The third monitor can be a small TV or laptop display, or it can be an actual

pinball score display device just like the real 1990s machines used.

Most virtual cab builders creating full-sized cabinets use the three-monitor setup. It provides the most realistic rendition of modern games that had speaker panels in the real machine, and it also gives you a good place to put the audio speakers (which is one of the big reasons the real machines adopted this design in the first place).

The **two-monitor setup** dispenses with the speaker panel and uses a single large monitor to fill the whole backbox.

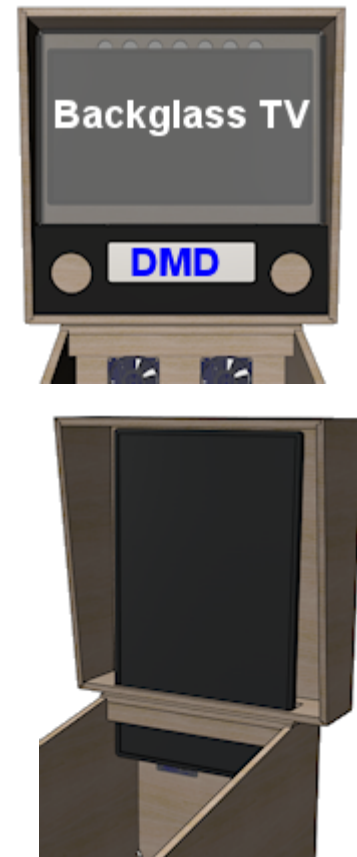
The advantage of the two-monitor design is flexibility. Classic tables from the 1960s and 1970s had larger backglassess that filled the entire backbox area, and the full-size monitor lets you display these older backglassess more realistically. A three-monitor setup has to squeeze the older, taller backglassess into a shorter area, which can distort the artwork. And you can still play modern games that had speaker panels originally, since the software can display a graphic rendition of the speaker panel on the screen.

Two-monitor setups are less common in full-sized virtual cabs, but you might be drawn to this design if you're especially fond of older tables from the "EM" (electro-mechanical) era. The artwork on those older tables can't be displayed as nicely on a three-monitor setup.

To help you decide, let's look at how various generations of real machines configured their backboxes.

Backglass styles through the years

Early pinballs displayed the score by lighting up individual point counter lights on the backglass. Pinballs in the 1960s and early 70s used mechanical score reels, which were positioned in little windows in the backglass art. These changed to 7-segment digital displays (similar to early pocket calculator displays) in the mid 70s, but they kept the same basic layout, with the digital displays positioned in the same little windows in the artwork that the mechanical reels had occupied.





Examples of pinball score display styles through the ages: point value lights (1940s-50s); mechanical reels (1960s); 7-segment digital displays (1970s-80s); dot matrix displays (1990s)

The biggest change came in the late 1980s, when Williams split the backbox between the glass artwork and a separate speaker/display panel. This arrangement had some major advantages, so it quickly became the standard. For one thing, it provided a good place for speakers. Pinball makers were doing everything they could to keep up with the competition from video games, and part of that was replacing the old bells and chimes from the electro-mechanical days with digital sound effects. Hiding the speakers inside the cabinet didn't make for very good acoustics, so Williams decided to dedicate some of the backbox area to speaker grilles. That meant sacrificing some of the artwork area.

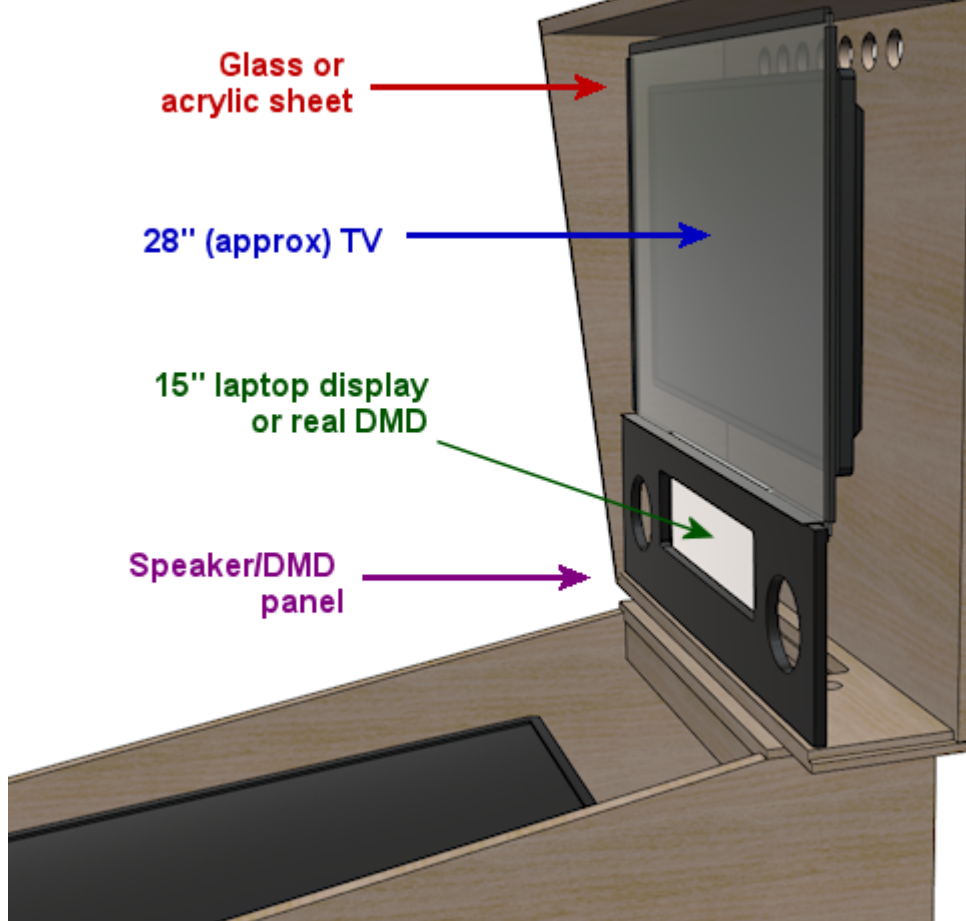


Early examples of the split design with separate backglass and speaker/display panel (1987). The lower panel is a separate piece that contains the score displays and a pair of speakers. These early games used 14-segment alphanumeric displays; later games used a single large 128x32 dot matrix display.

The three-monitor configuration in detail

Most people building full-sized cabs opt for the three-monitor setup. Part of the reason is practical. It's easier to make everything fit, it gives you a good place for the speakers, and it's easier to find a suitably sized TV. The other part is the aesthetics: it looks more like a real machine from the modern era.

You'll probably gravitate towards this design if you're generally more interested in modern tables than classics from the 1970s or earlier. This is also the most straightforward design if you plan to use a dedicated dot matrix display (DMD) device, since it replicates the setup of the real pinball machines that used those devices. It would be possible to fit a DMD somewhere else if you really wanted to, but the main motivation most people have for using a real DMD in the first place is to make the cab look more authentic, so unconventional placement would somewhat defeat the purpose.



Standard three-monitor setup, with TV for backglass and separate display for DMD area. The third display can be a small TV or laptop display panel, or it can be a real pinball DMD device. The clear glass or acrylic cover in the shape of a standard translite is optional; it's there to better replicate the appearance of a real machine, and to help hide the edges of the TV, which won't quite perfectly fill the space.



A 16:9 TV is a close (but not perfect) match to the standard proportions for modern translites. It leaves just a little extra space above and below.

The three-monitor setup is great for reproducing the backglass art for modern

machines that had the speaker panel setup in real life. It's not as good for older machines without speaker panels, since their backglass art was almost square. Displaying square artwork on a 16:9 TV requires a vertical squeeze to make it fit. This distorts the geometry a bit, as illustrated below.



Original proportions of classic backbox artwork (left); squeezing it onto a 16:9 monitor (right)

As you can see, full-height artwork is a little distorted by the vertical squeeze. I'm personally not too bothered by it on my own three-monitor setup, but then again I mostly play newer tables. If you play a lot of older games and you think the distortion would really bother you, you might consider the two-monitor option described later in the chapter.

Sizing the TV

The standard size of a modern backbox is about 27" by 27" on the inside. This leaves room side-to-side for about a 30" widescreen TV. Unfortunately, it's not possible (currently) to buy a 30" TV. The closest options I've seen are 28" and 29". If you can find a 30", it should be a perfect fit, but failing that you should look for a 29" or 28".

The next size up is 32", but this is too wide for a standard backbox. (You can't even fit a 32" with kludges like thinner side walls or routed slots in the side walls, since most 32" TVs are a hair wider than the *outside* dimensions of a standard backbox.) The only way to make a 32" fit is to build a custom backbox that's about two inches wider than standard. For some cab builders, it's worth doing this to get a perfect fit to a common TV size. If you go this route, keep in mind that you'll also need to a custom speaker panel and translite to match the special width.

The proportions of the standard translite space are approximately 16:10 (width to height). That's very close to standard 16:9 TVs - just a hair taller. Some computer monitors come in 16:10 ratios, so you might check to see if you can find something like that in the 29" or 30" range, but it's unlikely. Fortunately, 16:9 is so close to the real aspect ratio that you don't have to worry about distorted geometry in the artwork. The only reason to prefer a 16:10 monitor is that it would more completely fill the available space.

Score panel options

The three-screen configuration obviously requires that third screen, in the score panel window in the speaker panel.

This third screen can be another video display, or it can be a dedicated DMD (dot matrix display) device like the ones used in the real machines from the 1990s. Furthermore, it can be *exactly* like the ones used in the 1990s - specifically, a certain type of monochrome plasma display, which is still being made - or it can be a similar device with the same pixel layout that uses LEDs instead of plasma.

We'll look at in detail in the next chapter, Chapter 9, Selecting a DMD Device.

The two-monitor configuration in detail

So far, we've only looked at the "three-monitor" setup. Way back at the top of the chapter, we said that there was another option, *without* the speaker panel, where you use one large TV to fill the entire backbox space. This is known as the "two-monitor" configuration, because you end up with two TVs in your system (one for the main playfield, one for the backglass). Let's finally take a look at this alternative.

This is arguably the more flexible option, although it's also the more difficult of the two to set up. It's more flexible because it does a better job at reproducing older machines with full-height backglasses at the correct proportions, but it doesn't leave out the newer machines either, since it can show a newer machine's speaker panel "virtually" with on-screen graphics. The virtual rendition of a speaker panel obviously can't look quite as realistic as an actual speaker panel, but it does the job. If you're a big fan of classic tables from the electromechanical era, where the backglass art filled the whole backbox space, you might be willing to live with the fake speaker panels on modern machines in exchange for proper artwork proportions on classic tables.

But there are some major drawbacks. One is that it doesn't leave room for speakers. The real pinball makers adopted the separate panel design in part because it allowed the speakers to be exposed, which makes them sound better. You'll have to find another place for your speakers if you go the two-monitor route. You might be stuck (as the older real machines were) with placing the speakers somewhere inside the cabinet, which might somewhat reduce the audio quality.

The other big challenge is that it's impossible to buy a TV with exactly the right proportions to fit a backbox. The modern standard backbox is roughly square, about 27" wide by 27" tall (on the inside). Virtually all TVs and computer monitors sold today have 16:9 aspect, and the ones that don't are mostly even wider.

The solution that most two-screen cab builders use is to turn the TV sideways, so that the long dimension is vertical. This will make the TV too tall for the backbox, but you can cut an opening in the floor of the backbox and tuck part of the TV through the opening and into the main cabinet. This is illustrated below.





Typical two-monitor setup. The TV has to extend into the cabinet through the "neck" in order to fit vertically.



Proportions of the display in a two-monitor setup. The monitor can't fill the whole width of the backbox because it has to fit through the neck into the main cabinet.

You should be aware of a big drawback of this arrangement: you won't be able to fold the backbox down without removing the TV. On real pinball machines, the backbox is designed to fold down so that it lies flat on top of the cabinet, to allow for easier transportation. With the TV arranged like this, you'll have to take out the TV if you want to fold down the backbox. And you really should fold it down before transporting it, because there's a big risk of breaking something during transport with the backbox up, due to its weight and the leverage it has in that position.

TV size

Considering only the backbox inside width of 27", the ideal set would be about 53". But that won't work because of the need to tuck the end of the TV into the main cabinet. So your actual size constraint is the main cabinet width. This means that **your maximum backbox TV size is exactly the same as your main playfield TV size**. For a standard width cabinet (20.5" inside width), you can use a 39" or possibly a 40" TV; for a widebody cabinet (23.25" inside width), you can use a 45" TV.

This will leave some leftover space on either side of the TV if you use the standard modern backbox dimensions. You could simply fill this area with a black border or

decorative graphics.

There's another alternative, though. If you're enough of a fan of older EM machines to want a two-monitor setup in the first place, I'd suggest adjusting your cabinet plans to use a narrower backbox to fit the monitor. This will actually make your whole cabinet better fit the classic theme, since narrower backboxes were common until about the early 1980s. For example, the classic Gottlieb "wedgehead" style of the 1960s had backboxes about the same width as the cabinets. A 39" TV will fit these backboxes perfectly.



9. Selecting a DMD Device

Most real pinball machines from the mid 1980s to present use a split backbox setup, with a backglass at the top showing the game's theme artwork, and a separate panel at the bottom containing a scoring display and the audio speakers.



Typical WPC backbox layout. The bottom 1/3 is the speaker panel, containing the audio speakers and a dot matrix display (DMD). Games made up until about 1995 used the style shown here, with silkscreened graphics on the front of the speaker panel. Later Williams games used a more generic black plastic panel without any graphics apart from a Williams or Bally logo. The upper 2/3 is the backglass, displaying backlit still artwork for the game.

The backglass portion on most machines from the 1990s is just some static artwork, usually a "translite" (a screen-printed piece of plastic film stuck to the back of a clear sheet of glass), lit from behind.

The scoring window changed over the years. In the 1980s, they used segmented numeric displays, and later alphanumeric displays, like on old pocket calculators. In the early 90s, they switched to monochrome dot matrix displays ("DMDs"), typically 128x32 dots. The DMDs could display full graphics, although 128x32 monochrome pixels is obviously very coarse by today's standards.

Many virtual cab builders follow the 1990s design, except that they replace the backglass with a video monitor. This means that you need a separate display device in the DMD area, which is why this design is usually called the "three-screen" cabinet: you have one screen for the playfield, a second for the backglass, and a third for the DMD. The third screen can be an actual 128x32 monochrome plasma DMD, just like in the the 1990s originals, but most cab builders these days substitute a small video panel instead, since that's cheaper, easier to set up, and more versatile.

This section looks at the options for the third screen, to help you decide which type to use, and offers some pointers for buying the equipment you decide on.

Overview of DMD technologies

The DMDs in the original 1990s pinball machines were monochrome plasma displays, 128 pixels wide by 32 pixels high. That made for very large pixels that you could see individually. This visible "dot" structure, and the particular amber color of the plasma, gave the displays a distinctive appearance that many people now see as a defining feature of this generation of pinballs. To a lot of people, it doesn't feel like real pinball if it doesn't have the amber dots.

You can still buy the original plasma panels, and it's even possible to use them in a virtual cab (although, as of this writing, there are no commercial interface kits available to facilitate this). There are also newer display technologies that can be substituted into the score panel to achieve similar looks, with some modern improvements.

The first newer alternative is LED-based 128x32 dot matrix displays. LED displays are available with the same pixel pitch and layout as the original plasmas, so they can serve as close substitutes. LEDs don't perfectly replicate the nuances of plasma's visual effect, which has a soft, analog, neon feel to it that some people find charming. LEDs are crisp and bright but can seem a little harsh in comparison. But LEDs definitely share some of the more important positive properties of plasma, particularly high brightness and wide viewing angle. LEDs are also cheaper than plasma and longer lasting, so collectors of the real machines often replace defunct plasmas with LED panels when repairs are needed. LEDs are also being used on many new titles being shipped today, so plasma is no longer the sole "original equipment" on real pinballs. Stern no longer ships new games with plasma displays at all; they switched to LED DMDs in 2013.

The original LED DMD panels were monochrome (available in a variety of colors, including something approximating the distinctive plasma amber), but panels are now widely available with RGB pixels, which can display full-color graphics.

The other newer alternative to plasma is to use an actual video display, typically an LCD panel. For a virtual cab, an LCD panel is easier to set up in terms of software, since it just looks like another video monitor as far as Windows is concerned. Pinball simulators will happily simulate the look of a plasma DMD on a video display by drawing large amber dots to simulate the 128x32 pixel structure. Of course, an LCD panel can't perfectly reproduce the brightness or viewing angle of a plasma, but it can at least do a passable impression of the appearance.

A 15" diagonal 16:9 LCD screen happens to fit the width of the standard DMD opening in the pinball speaker panels. It's a trifle taller than the standard panels overall, but since it sits behind the panel, that's not typically a problem, as it's hidden behind the translite, which sits directly on top of the speaker panel.

Recommendations

For a virtual cab, you can in principle use any of these technologies - an original plasma DMD, a monochrome LED DMD, a full-color RGB LED DMD, or a video display. (Although what you can actually buy right now is somewhat more limited.) The tradeoffs are complex, but it mostly comes down to your priorities:

- If you have fond memories of the 1990s machines, and you want to match that look, a plasma display is the way to go. Plasmas are the authentic equipment, so they'll look exactly right. A plasma is also quite bright, so some people like the way it becomes part of the "light show". The downsides are that plasma displays are expensive and fairly complex to set up. They require a special high-voltage transformer as the power supply; VirtuaPin sells an appropriate transformer, so it's at least easy to source, but it adds to the complexity for installation. Another downside to plasma devices is that they fade as they age

and eventually wear out, although I believe their longevity is a function of powered-on time, so you can probably expect a plasma to last a very long time in home use. I have several real pinball machines with original plasma displays that have been in home use for over 25 years, and I haven't had one exhibit any signs of fading or failing yet.

- If you want to replicate the 1990s look but want to reduce the complexity a bit, a monochrome LED is a good choice. These look very close to the plasmas - they're even brighter, and you can even get them in a fairly close color match to that special plasma amber if you want, as well as in a range of other colors. They're a bit simpler to set up than plasmas because they don't require the special power supply. They also have a longer reliable service life than plasma. They're more complex to set up than a video display, though, and more expensive.
- If you like the "dots" look of the 1990s machines but want to add full color support, consider an RGB LED. These are slightly more expensive than the monochrome LEDs, and they're about the same in terms of setup complexity.

I was really excited when the RGB LEDs first came out, because I thought they were going to be the perfect combination of the original plasma look and modern full-color flexibility. But I'm sorry to report that the reality isn't that simple. I know some people are going to hate me for saying this (particularly people selling RGB DMDs!), but I actually think a video display does a more convincing job of replicating the "dots" look than an RGB LED. The problem is that the sub-pixel structure on the RGB LEDs is way too obvious; it makes the individual dots look too small. It's very noticeably different from the plasma and monochrome LEDs. That's a first-hand opinion, too: I have machines with both kinds of displays at home, and to my eye the video display looks more like the real thing.

- If you're not attached to the idea of using 1990s-era equipment for its own sake, a video display is the best overall option. It's cheaper than any of the 128x32 DMD options, it's easier to set up, it's more flexible, and to my eye it actually does a better job of re-creating the "dots" look than the RGB LED displays do.

The only drawback of a video display (other than that it's not authentic 1990s pinball equipment, which you might or might not consider a drawback) is brightness. The plasma displays are quite bright, and a monochrome LED is even brighter. (RGB LEDs are a mixed bag on this score because of the sub-pixel structure; brightness depends quite a bit on the color being displayed at any given pixel.) From comparing my own machines with different display technologies, though, I think this is often overblown when people talk about it on the forums. Side by side, they're really not that different. And I think when you compare the overall visual quality, an LCD video panel has the edge.

In terms of flexibility, a video display can do both "dots" and full-resolution graphics. The "dots" look can be easily simulated on a full-res display, and all of the pinball software is set up to do just that, because it's all written primarily for desktop machines where video displays are the only thing going. What a video panel can do that a 128x32 DMD can't is display high-res graphics when it's not displaying "dots". For example, when playing an EM game that doesn't use the score window at all, you can use it to display added game graphics or manufacturer logos at full resolution. Video panels also look much nicer when playing 1980s "alphanumeric display" games, because they can accurately simulate the 14-segment display look.

Video is also the most compatible option. Every pinball program for Windows

naturally works with a video display, since that's just how Windows works; support for a DMD device has to be intentionally added on by each program's creator (or hacked in by reverse-engineering, which the pin cab community has successfully accomplished with several commercial titles).

When I first started on my virtual pin cab project, almost everyone building cabs felt that plasmas were the Cadillac of scoring displays, worth almost any amount of extra cost and extra trouble to set up. But I think this has completely reversed in the time since then, because the real pinball world has largely moved on to more modern technologies. These days, pinball machines you might see in public places use such a mix of dot matrix and video displays that both seem perfectly "real" now. Some of the newer Stern titles are shipping with DMD-sized video displays as original equipment, and Jersey Jack Pinball's entire line uses large video monitors in place of the DMD panel. You even see lots of classic 1990s machines retrofitted with full-color video displays, thanks to ColorDMD, a company that makes drop-in replacement displays for the old machines. So I expect that many cab builders starting new projects now and in the days to come will be less fixated than earlier cab builders were on the idea that the plasma DMDs were the only "real" pinball displays. There's definitely a lot of nostalgia value to the old plasmas, but overall I've come to think that video is the best option.

Purchase options

At one point, it was possible to buy any of the display technologies mentioned above - plasma, monochrome LED, RGB LED, or video. But the buying options have become a lot narrower lately. The PinDMD v2 doesn't appear to be available any longer, and that was the only readily available way to hook up an original plasma display or a monochrome LED panel.

So at the moment, there are two options: video, or RGB LED.

LCD video panel

If you plan to use a video panel for the score display, the best fit is a 16:9 panel, approximately 15.5" diagonal. This is just about a perfect fit for the 13.6" width of the standard DMD opening in a speaker panel. A panel of that size is just barely taller (by about a centimeter) than the standard speaker panel's outside dimensions, but that's typically not a problem, because the excess height is easily behind the translite panel, assuming you're using one.

A few TVs are available in this size range, but I'd recommend against those. They tend to use low-quality LCD panels. The much better solution is to use a laptop display panel.

You can buy replacement laptop LCD panels in this size range on eBay or Amazon. These panels come bare, with no interface electronics, because they're sold for repair work where you only need to replace the panel and nothing else. This means that you have to buy a separate piece of electronics, called a video controller, that serves as the interface between the panel and the PC video card.

To find these parts, start by searching eBay for "15 wuxga". You should find a number of matches, usually listed as replacement parts for Dell, HP, Acer, and other laptop brands. You should narrow the list to panels that specify 1080p or, equivalently, WUXGA (1920x1080) resolution, and a screen size of 15.5 or 15.6 inches. The price range for these panels as of this writing is about \$50 to \$100. The matches you're looking for are just bare laptop display panels - an LCD screen in a thin metal shell. They'll look something like this:



Don't try to choose a specific panel yet; just keep the search results ready. The next step is to find a video controller that works with one of these panels. eBay doesn't provide any tools to help with this, so you'll have to do some manual searching. Open a new eBay search window. Go down your list of panels. For each one, find its model number in the listing and type it into the search window, adding "controller". For example, if you find a panel with model number LP156WH4T, type "LP156WH4T controller" into the search box. If you're lucky, that will turn up a few hits with the model number in the title. Be sure the model number is actually in the title or is explicitly mentioned in the listing as a compatible model. The controllers will usually look something like this:



If you don't have any luck, or you're not sure you found the right match, I'd recommend picking a panel that looks good and contacting the seller to ask which controller to use. The seller should be able to point you to the right device. Most of these panels use similar control interfaces, so you don't actually need a controller designed especially for your panel. Sellers list them for specific panels simply because they know people like us are searching for them that way! Technically, you just need a controller that matches the interface type on your panel, but the ads don't usually list enough information to find them that way, so a model number search is the only way to be sure.

Pay attention to connectors. Most of the interface boards will have a VGA input and either a DVI-D or HDMI input. If you've already picked out a graphics card for your cabinet PC, be careful to pick an interface board that has a connector matching an available output on your graphics card, taking into account the outputs you'll be using for your main playfield TV and backbox TV.

How do you know if a panel is good in terms of video quality, reliability, etc.? You're not going to find reviews (professional or user-written) for any of these OEM parts, so it's a bit of a crapshoot. Fortunately, laptop panels in this class have gotten to be good enough that you should be okay with anything that meets the specs. Do pay attention to the resolution, though: the WUXGA laptop displays seem to be uniformly good, but the lower res displays are uniformly bad.

One note on setting up your new panel: be aware that the control board might support more resolution modes and refresh rates than the panel itself does. Many of

the modes that the controller allows you to select with the Windows control panel might simply not work with the panel or might produce poor-quality video. When you first set up the panel in Windows, make sure you select the screen size (resolution) and refresh rate that exactly match the panel's physical design. That might take some trial and error, since eBay OEM parts don't usually come with any documentation. If the display looks fuzzy or distorted, or doesn't show anything at all, try other refresh rates to see if you can find one that looks better.

RGB LED

If you decide on to use an RGB LED dot matrix display device, there are two ways to accomplish it: you can buy one commercially, or you can build one yourself using DIY plans that some pin cab enthusiasts developed and published.

RGB LED - commercial

VirtuaPin sells the PinDMD v3, a full-color (RGB), LED-based, 128x32 dot matrix display. The display panel has the same physical dimensions and dot pitch as the original plasma displays in the 1990s machines. It's about \$270.

This is a turn-key commercial kit, so it's relatively plug-and-play. It uses a USB device to interface to the PC. It requires some software setup; instructions are included, and VirtuaPin offers warranty support.

RGB LED - semi-DIY

Pin2DMD is a DIY project for building an RGB DMD panel from parts. The site provides a parts list and assembly instructions, as well as software for a microcontroller to interface to the PC and run the display. The prices for the parts vary, but at a guess they'll total about \$100.

Note the confusingly similar name: Pin2DMD is the DIY project, and PinDMD v3 is the commercial product above.

The Pin2DMD site includes software to install on a microcontroller board (one of the parts that goes into building the project) that interfaces with the PC and runs the display. However, note that the software is *not* open-source, and requires payment of a license fee.

The closed-source software makes me hesitate to recommend the project. It's supposedly "DIY", but given that you don't have any control over the software or any ability to change it to suit your needs, I think "DIY" is actually a negative in this case. You have to do the assembly yourself, you don't get any warranty support, and you don't even get any control over the final product. Open-source projects have the first two drawbacks, but they make up for it by giving you full control to customize and expand. You don't get that here; you just get the bad aspects of DIY and the lack of control of commercial products. But I guess you can at least save some money vs the retail version.

Plasma panels

Plasma doesn't appear to be an option at the moment. VirtuaPin formerly offered the "v2" PinDMD, which was a monochrome of the PinDMD v3 device mentioned above that worked with your choice of monochrome 128x32 LED panels or the original plasma panels used in the 1990s machines. But that product doesn't appear to be available anywhere as of this writing.

You can still buy the bare Vishay plasma panels from VirtuaPin, along with the special 80V/100V transformers needed to power their high-voltage sections, but VirtuaPin doesn't sell anything that would let you hook it up to a PC. I don't know of any other

commercial or DIY options for connecting these.

If you're an experienced software developer with some hardware knowledge, you could design your own controller using one of the inexpensive ARM-based microcontroller boards, such as a Raspberry, BeagleBone, or one of the STM32F series boards. The software involved is actually very simple: you just need to consume USB packets from the PC and send out a clocked serial bit stream to the plasma device, 1 bit per pixel. The electronic interface is documented in the Vishay data sheets, and it will be immediately recognizable and straightforward to anyone who's done any device interface work with a microcontroller before. If you do create such a project, please publish it as open source, and let me know about it so I can include here.

Monochrome RGB panels

As with the plasma panels, you can buy monochrome RGB panels as components, but there's no software interface to the PC available. The panels are available from a few after-market pinball suppliers who sell them as drop-in replacements for dead plasma displays in real pinballs. Since they're specifically designed as drop-in replacements for the Vishay panels, their electronic interface is identical, so any solution you can find that will work with the Vishay panels will work equally well with these. As requested above, please let me know about any solution you develop or find for this and I'll add it here.

10. Designing the PC

The core of a pin cab is a PC running Windows. You could theoretically build a pin cab around a Mac, an iPad, a Raspberry Pi, or just about any other sort of computer. But for our purposes in this guide, the only real option is a Windows PC, because that's where all of the software runs.

I've observed that most pin cab builders like to start their projects by building the PC and setting up the software, before they've even started thinking about what's needed to build the pinball machine body that'll house it. This is a natural first step for most of us, because most of us know our way around PCs at least a little bit - the way that most people discover the pin cab world in the first place is through the PC pinball simulation community. It's also an attractive place to start because you can see some immediate results, before getting into the more daunting parts of the project.

Off-the-shelf or custom build

The easiest and most obvious way to get a PC is to buy one from a retail PC maker, or even re-use one you already have. But most pin cab builders come from a PC gaming background, so you probably already know enough about PCs to know the benefits of building one yourself rather than buying off-the-shelf. If you've built your own PCs in the past, you know what's involved. If you haven't built one before, you might be surprised at how easy it is. Modern PCs snap together out of components practically like Lego blocks. The hardest part is often the shopping, since there are so many options out there.

The big benefit of building your own PC is that you get to pick exactly what you want for each component. The retail PC makers usually give you a few options for CPU speed, hard disk size, graphics card, and so on, but they're usually pretty limited choices from a small pre-set list. If you build your own, you can choose exactly what you want from the whole universe of available products in each category.

The rest of this chapter proceeds from the assumption that you're going to build a custom PC, because that's what most pin cab builders do. But that's not a must; if you're not comfortable building your own PC, you can definitely build a perfectly good pin cab around a retail PC. If you go that route, I'd suggest you focus on PCs that are specifically designed and marketed for gaming. PC pinball is fundamentally a video game, and it benefits from exactly the same hardware upgrades that mainstream video gamers need. Pay particular attention to the graphics card: that's the hardware element that makes the biggest different for PC pinball performance. You might find the material in this chapter helpful even for picking out a pre-built PC, just for the background knowledge of what to look for and which elements are the most important to pin cab performance.

Performance considerations

I can't give you any hard numbers for performance metrics, since things change too quickly in this business and any benchmarks I quote would be obsolete in a couple of months. I can offer some general advice, though.

The first bit of advice is that you should consider the virtual cab PC to be a gaming PC. That might seem obvious, but my point isn't merely that you're going to use it to play games, but rather that "gaming PC" is a special category of PC. The thing that makes a gaming PC different from a run-of-the-mill home or business PC is upgraded performance, particularly for graphics. Gamers use special disks, special memory,

and most of all special graphics cards.

The second bit of advice is that you don't have to take this idea of upgraded performance to its logical extreme. You do need good performance, but you don't need the absolute best performance available. Pinball emulation is demanding, but it's not as complex as the latest "triple A" video games at any given time. My rule of thumb is that you should look for the "second best" in most of the product categories. Survey what's available, and don't buy the most expensive thing you can find; focus your attention on the second price tier. Products in that second tier are usually only slightly less capable than the top-tier products, but much cheaper - you often see crazy things like 90% of the performance for half the price. The gamers who want the *very best* are willing to pay, pay, pay for it. So products in that second tier often offer a much better balance between price and performance.

To a first approximation, the CPU and GPU together determine your machine's overall performance. And of the two, the graphics card is generally the more important. These are the parts you should pay the most attention to when researching what to buy.

Other components - motherboard, memory (RAM), disks, USB controllers - also contribute to performance, but to a much lesser degree. How much should you worry about those? If you talk to serious video gamers, they'll tell you that every element is critical, down to the military-grade titanium screws holding their ballistic carbon-fiber cases together. That's true as far as it goes, but "extreme gaming RAM" and the like will only contribute a few percentage on most systems. Most people can't perceive that kind of difference in actual use; you'd only know it's there if you measured it with benchmarking tools. If it's important to you to build the fastest system possible, then by all means do so; that can be fun in its own way. If your main focus is pinball rather than PC benchmarks, I'd focus my research time and cash budget on the CPU and GPU, and I wouldn't go too far out of my way seeking the optimal choices for the other components. Just look for parts from reputable manufacturers that fit the specs you need.

Operating System

Recommended: Windows 11, 64-bit, Home edition. Windows 10 is also still a good choice.

Windows is the best operating system option for a pin cab PC, because almost all of the popular pinball software runs only on Windows. This is starting to change, with some of the open-source software being ported to Linux and MacOS, but it will be a while before the ports are as solid as the Windows versions, and most of the commercial games will probably never be ported.

Which version

I'd recommend the latest, currently Windows 11. The main reason is that Microsoft only offers full updates to their DirectX technologies (their gaming technology layer) on the current OS version at any given time. This means that newer games will increasingly be unable to run on the older Windows versions; if you want to be able to run the latest games, you pretty much have to have the latest Windows.

Older versions: As of this writing, Windows 10 is still well supported, but all of the older versions are now out of support, so they're no longer receiving updates from Microsoft. As I'm sure you've heard from many other people, the big concern when Microsoft terminates updates is that security bugs in the OS won't get fixed, so it will become increasingly vulnerable to malware. For a pin cab PC, I think the termination

of DirectX updates is an even bigger problem.

Which edition?

The "Home" edition of Windows is fine for a pin cab. You can buy the more expensive "Pro" edition if you prefer, but the added features in the Pro editions are intended more for business users. I don't think there's anything in Pro that's important for a typical pin cab.

32-bit or 64-bit?

Easy: Use the 64-bit edition. The 32-bit version of Windows is only for old hardware with CPUs from about 2002 or earlier. Every Intel and AMD PC CPU you can buy today is 64-bit. The 64-bit version of the operating system takes full advantage of the CPU's capabilities, and is still fully compatible with 32-bit application software.

Emulation and virtualization options

Nope. Don't even consider it. Even though it's technically possible to run Windows as a guest operating system using VM software on Linux and MacOS, it's not a viable option for gaming software. 3D gaming performance on virtualized hardware is uniformly unacceptable. The same applies to Wine (a Windows API emulator on Linux).

Hardware components

Here are the PC components you need to assemble the computer that runs a virtual pin cab.

CPU

Most people start planning PC builds with the CPU, because other choices hinge on which CPU you choose.

This is obviously an area where any specific product recommendations I make will quickly become outdated, but there are some guidelines that seem pretty stable over time.

First, any current Intel or AMD CPU that's in the middle of the performance range or better should handle most of the pinball software easily.

Second, most gaming software, including pinball simulators, generally benefits more from what they call "scalar performance" than from adding more "cores". Scalar performance refers to how fast each individual CPU core runs, and it's roughly proportional to the clock speed, as long as you're comparing chips that are from the same generation.

Adding more cores is usually less beneficial for pinball software than scalar performance, because most game software has a single critical path (that can only run on a single core at a time) that determines the overall performance. However, more cores are better to a point; I'd go with a CPU with at least four to six cores.

Third, the CPU feature called "hyperthreading" is generally considered not very useful for gaming software. Hyperthreading is the main distinguishing feature of some of the highest-end Intel and AMD CPUs, and it improves performance for many types of software, but most gaming software is designed in a way that hyperthreading doesn't do much for it. It might not be worth the cost bump in a pin cab.

CPU performance is always a hot topic in the gaming community, so it's easy to find

lots of detailed performance data on gaming-related Web sites. Several sites run extensive benchmarking suites and publish their results. You should give the most weight to tests for gaming performance, since pinball simulators are similar to other video games in the way they use the machine's hardware resources.

CPU fan

Most modern CPUs require a special fan mounted directly on top of the chip. If you buy your CPU in retail packaging, it usually includes a suitable fan. However, some vendors sell unpackaged "OEM" versions intended for use by business buyers building systems for resale, and these usually don't include anything but the bare CPU. In that case, you'll need to buy a CPU fan separately. These can be found on Newegg and other sites that sell components by using a search term like "i5 fan". Check the specs on the options you find to make sure your specific CPU type is listed, since these fans usually have to match the exact shape and size of the chip.

Motherboard

The motherboard is the main system board with all of the core electronics, and connectors for all of the add-in cards, disks, and input devices.

Choose a CPU before looking for motherboards. Any given motherboard only works with specific CPUs. Once you know the CPU you're going to use, you should be able to find suitable motherboards by searching the Web for "xxx motherboard", where "xxx" is your CPU type. Use the detailed CPU part number, like "i5-7600k".

I've had good results with motherboards from Gigabyte, but several other manufacturers make good motherboards as well.

For a pin cab, your needs from a motherboard aren't very complex. Here are the main features I'd look for:

- Must have: Compatibility with your chosen CPU
- Must have: At least one fast expansion slot for a graphics card, typically PCI Express x16 (as of this writing).
- Must have: At least two additional expansion slots, in case you want to add a sound card, Wi-Fi card, USB card, or any other add-ins.
- Must have: Memory slots for at least 8GB of RAM. (This is almost a given; it's hard to find a board *without* at least this much capacity these days.)
- Nice to have: on-board Ethernet port. This is standard on nearly all modern motherboards. Wi-Fi is less important, because you might not be able to use a built-in antenna effectively; the walls of a pin cab are thick enough to significantly block the signal. An external antenna is usually better if you want Wi-Fi on the cab, and for that you'll probably need an add-in card or an external USB Wi-Fi adapter.
- Nice to have: integrated audio. Nearly all modern motherboards include audio outputs. This isn't required, though, as you can add a sound card via an expansion slot if needed.
- Nice to have: USB 2 **and** USB 3 connectors. Some older USB devices don't work well with USB 3 ports, so it's helpful to have both types in case you need a USB 2 port for some devices. This isn't required, though, since an external USB 2 hub can serve the same function.

Performance considerations: Not really an issue, unless you're looking to build an extreme gaming system. A motherboard designed for a particular CPU is almost always based on the Intel or AMD chipset mated to that CPU, so you won't see a

huge amount of variation among different boards for the same CPU. If you're concerned about finding the fastest motherboard for your CPU, you can do some research on benchmark sites on the Web.

What about on-board graphics? Unimportant, because you'll need a separate graphics card whether or not the motherboard has its own built-in graphics. There might be exceptions, but all of the built-in motherboard graphics chip sets I've ever seen are low-end, suitable for business graphics, not gaming.

If the motherboard doesn't have on-board graphics, great, that's one less thing to worry about when configuring the BIOS. If it does have on-board graphics, as most modern motherboards do, it's still not a problem because you should be able to disable it in the BIOS setup. In fact, many BIOSes will do this automatically when they detect the presence of a separate video card.

Graphics cards

The graphics card is the most important component for game performance. It's even more important than the CPU for games, because it's actually a whole separate computer in its own right that does most of the computing work for displaying 3D graphics. Fast graphics cards are capable of drawing more complex images more rapidly, making for smoother game action.

You should wait until after selecting a motherboard to choose a graphics card, because you need a graphics card that matches the "slot" type on your motherboard. Your motherboard specs should tell you what kind of graphics cards it accepts; look for "graphics cards" or "expansion slots" in the spec sheet. For quite a while now, motherboards have been standardized on "PCI Express" slots for the graphic interface. These are quoted with a speed like "x16", so you might see "PCI Express x16" in the expansion slot list. Once you find that information, that tells you what types of graphic cards are compatible.

Graphics cards are available from many manufacturers, but most (regardless of manufacturer) use chip sets made by either Nvidia or AMD. The spec sheet should tell you the underlying chip set, and in fact, most cards from most brands include this information right in the name. For example, a "Gigabyte Geforce GTX 1050" is based on the Nvidia 1050 chip set. You'll start to recognize the chip set names if you shop around enough, since you'll see the same numerical designations over and over on different brands of cards. The performance of a graphics card is almost entirely a function of the chip set, not the brand, so you should see reasonably similar performance from cards based on a given chip set even if they're from different brands.

Which chip set? Check the forums for advice on current models. This is another area where something in the second tier of the current available performance range is usually a good choice.

Note that a 4K main monitor is more demanding than a regular HD (2K) monitor, so you should bias your search towards the higher end if you're planning on 4K.

Video memory: Video cards have their own on-board memory, usually 1GB or more on a modern card. The fastest type of memory has a type like "GDDR3" or "GDDR5". A higher number suffix indicates faster memory. Visual Pinball and other gaming software benefits from larger memory sizes with faster memory.

Display size and refresh rate: Any modern video card should be able to drive a 1080p main monitor and a couple of additional smaller monitors. (1080p or even 720p is perfectly adequate for a backglass TV.) A higher-end card might be needed if you're using a 4K main monitor.

Outputs/connectors: Be sure you have enough outputs for all of the monitors you plan to connect, taking into account the playfield TV, the backbox TV, and the score display (DMD) TV, if you're using that.

Most higher-end graphics cards offer several output ports with different types of connectors. You can almost always use all of the outputs simultaneously to drive multiple monitors. This lets you use a single graphics card to drive all of the TVs in your system.

I'd recommend finding a card with at least two of the following connectors, in any combination: HDMI, DVI-D, Display Port (DP). All of these types can be connected (using passive adapters) to HDMI inputs, which is what you'll need on almost any modern TV. As long as you have two ports of these types, you should have no problem connecting two TVs to the card.

If you're planning to also use a third display for the DMD area, you'll need a third output for that. A VGA or DVI-D connector will usually work for this third output, since DMD monitors are usually implemented with laptop displays or small desktop monitors. Most video cards have a VGA output in addition to one or more of the more modern connectors listed above, so this is fairly easy to find.

If you're going to use a real pinball DMD instead of a small video display, you **won't** need to connect that the graphics card. Real DMDs aren't video devices, so they don't connect to your graphics card; they connect instead to a special external controller via USB.

You should check the specs to confirm that the card you're considering can handle the two or three simultaneous outputs you plan to use. Nearly all modern graphics cards allow this, but it's worth checking to be sure.

Port compatibility: You don't necessarily need an exact match between the output port types on your video card and the input ports on your TVs and monitors. Many of the port types are electrically compatible with each other, meaning you can connect them with a simple cable that has the right plug on each end.

The following combinations of port types are compatible. The only requirement is a cable with the corresponding connector type at each end. These are relatively inexpensive and can be easily found online.

TV IN	Video Card OUT	Compatible?
HDMI	HDMI	Yes
	DVI-D	Yes
	DisplayPort	Yes
DVI-D	HDMI	Yes
	DVI-D	Yes
	DisplayPort	Yes
DisplayPort	DisplayPort	Yes
VGA	VGA	Yes
	DVI-I	Yes

Two cards for two monitors: Not advised. It *seems* like two cards would be better than one - more hardware is always faster, right? But in practice, two cards are actually *slower* than one. Everyone on the forums who's tried this has had the same results: you get lower frame rates, more stutter, and more lag with multiple video cards.

The technical reasons for this are unclear (my wild guess is that it's due to increased PCIe bus contention). Without understanding the cause, I can't rule out the possibility that *some* systems exist where two cards would go faster than one. But if there are, they seem to be the exception; many people have tried it and had poor results.

By far the best way that anyone has found to improve performance of the pinball simulators is to use a faster video card.

Using the motherboard GPU as a second video card: Not advised, for exactly the same reasons that you shouldn't add a second video card (above). Enabling the motherboard GPU is exactly the same as adding a second video card in terms of its effect on your overall system performance: you'll see lower frame rates, more stutter, and more lag if you enable the on-board GPU.

Memory (RAM)

I'd recommend at least 8GB of motherboard RAM. This is enough memory for Windows plus the pinball simulator to run comfortably without "swapping" to disk. More RAM is generally better - particularly for future-proofing, considering that Windows and other software tends to need more memory on every update. If you have the budget, you can install as much memory as your motherboard can accept.

The type of RAM chip you use must match the requirements for your motherboard. You can find the RAM type requirements in your motherboard's spec sheet, but it's usually easier to find the right chips by typing your motherboard's model number into a Web store's RAM search. Most online stores that sell RAM let you search for compatible chips by motherboard, narrowing the results to show only compatible products once you enter the motherboard information.

You'll probably be able to find many compatible RAM chips for your motherboard. These will be listed with a speed class like "DDR3-2000" or "DDR4-2133". "DDR3" and "DDR4" are essentially versions of the electrical interfaces, so your motherboard will probably accept exactly one of these types. The suffixes like "-2000" are clock speeds, so higher numbers are faster in terms of the bus clock. These numbers don't translate directly or linearly to overall system throughput, since there are many other factors besides the raw clock speed that affect the actual performance, but using higher-speed RAM will generally increase overall system speed. I'd recommend buying the fastest speed class that your motherboard supports, since the price differences between RAM types aren't usually dramatic, but you can let your budget decide, since the performance differences probably won't be dramatic either.

Note that you might see the "DDR" speed class combined with another class with a "PC" prefix, such as "PC3-16000". These are just different ways of stating the same information. Don't compare "DDR" speeds with "PC" speeds, since they're different systems - only compare "DDR" speeds with other "DDR" speeds, and "PC" speeds with other "PC" speeds.

In addition to the "DDR" speed class, you might see a series of other specs, such as "Timing 15-17-17-35" or "CAS Latency 15". These numbers are further details about the memory speed. Hardcore gamers try to optimize these, but I don't recommend worrying about them, because they represent very slight differences in speed that might not even be noticeable in actual use. The "DDR" speed class and the total

amount of RAM are much more important.

Hard Disk

The best type of hard disk for a virtual pin cab PC is an SSD, which isn't actually a "disk" at all, but serves the same storage function using flash memory instead of magnetic media. SSDs are much faster than conventional hard disks, especially for booting Windows and loading software. Booting Windows from an SSD typically takes ten or twenty seconds, compared with a minute or more with a conventional hard disk.

SSDs also smaller than convention disks, and they're essentially immune to damage from vibration or shock. The shock resistance is good for a pin cab since you'll want to be able to nudge the machine without worrying about damaging the disk.

The main drawback of SSDs is that they're more expensive than conventional hard disk per gigabyte. Fortunately, a pin cab doesn't need a very large disk, so most pin cab builders will find that a suitably sized SSD is well within a reasonable budget for the PC components.

How much storage do you need? Let's look at what you'll typically need to install on a pin cab PC:

- Windows operating system: about 20GB
- Visual Pinball: about 20MB
- Future Pinball: about 100MB
- VP and FP tables: varies, hundreds of MB
- PinballX (menu system): about 40MB
- PinballX media (table images, videos, etc): varies, hundreds of MB
- Web browser: 1GB
- Other software and utilities: varies, hundreds of MB

We obviously can't come up with an exact number here because the total will depend on how many tables you install. But we can still come up with a pretty good upper bound, since there are only so many tables out there (perhaps 1000 in circulation), and they're not all that big individually (perhaps 1MB to 20MB apiece). Even if you install all of the tables you can find, and even if you never delete the less interesting ones, you're probably talking about less than 20GB of total disk space required for them.

Adding all of this up, we come up with about 45GB. Realistically, you'll want to increase that figure to account for the inherent overhead in the way Windows uses disk space, and to leave some room for temporary files, downloads, etc. So I'd recommend an absolute minimum disk size of about 65GB, and preferably at least 120GB. But given current prices, I'd step up to about 250GB - that size is available for about \$100 US as of this writing, which makes it the best value in terms of price per gigabyte. 250GB is plenty of space for all current pin cab needs and leaves lots of space for future expansion. Depending on when you read this, the best value size might be even larger, so shop around to see what's available.

Power supply

Virtually all motherboards and disk drives are compatible with the standard "ATX" type of power supply. The only exceptions are motherboards designed for very small form factor machines, so as long as you're using a standard full-sized motherboard, you should be able to use any ATX power supply from any manufacturer.

ATX power supplies are so standardized that you don't have to worry about the types of plugs it has or the types of voltages it produces. These are all uniform across all ATX power supplies. The only thing that varies is the total power capacity, expressed as a number of Watts. You'll have to pick a power supply that produces at least the wattage required by your motherboard and other components.

You can determine your wattage requirement by adding up the power figures in the specs for your motherboard and video card. Those are the two components that draw the most power in your system. Be sure to pay attention to the video card, because it might require even more power than your motherboard does.

For example, if your motherboard spec sheet says that it requires 200W, and your video card requires 300W, you'll need a power supply that provides at least 500W. I'd add about 100W to the total you come up with from the motherboard and video specs, to account for the little bit of extra power that will be drawn by the disk and USB devices, so in this example I'd look for a power supply rated for at least 600W. The number you come up with here is just a minimum: you can buy any power supply rated at this number or higher.

Sound cards

Most modern motherboards have integrated audio. For a basic setup, this is all you'll need.

If you want, you can add a separate sound card that plugs into an expansion slot on your motherboard. This will let you set up a second, independent set of speakers on the added card, in addition to the main set of speakers attached to the motherboard audio outputs.

Why would you want two sets of outputs? Visual Pinball has a special feature that lets you take advantage of two audio systems to separate the "music" tracks from the playfield sound effects. Many pin cab builders set things up so that the music plays from the backbox speakers, the same arrangement as in the real machines from the 1990s, and the playfield sound effects play through a separate set of speakers located inside the cabinet under the TV. The playfield effects include the sound of the ball rolling and bumping into things, so it improves the simulation to have these sounds come from the direction of the playfield.

If you do want to install a separate sound card for the playfield effects, you don't need anything fancy. Any inexpensive sound card will do. Just make sure that it uses the same type of expansion slot that you have on your motherboard. For most modern motherboards, these will be standard PCI slots.

Case or caseless

Most pin cab builders house the whole PC inside the cabinet. This makes everything self-contained and adds to the illusion of a real machine. That's not a requirement, though: some cab builders simply put a regular PC on the floor next to the cabinet. This works, but it's not as nicely integrated, and you'll have to run several cables (video and USB) between the external PC and the cab. It's fairly obvious how to set that up, so we'll ignore that option and focus on how to set up a PC inside the cabinet.

There are two main options for mounting the PC components inside the cabinet. The first is to build the PC using a conventional tower case, and then put the case inside the cabinet. The second is to skip the case and mount all of the PC components directly inside the cabinet, attaching them to the floor of the cabinet or one of the inside walls. The cabinet itself serves as the case. Each approach has some tradeoffs.

The main advantage of using a conventional PC case is that it provides structural support to hold the video card and other add-in cards in place, and provides places to mount the disks. A case also provides physical protection from falling objects, and provides shielding for the radio frequency energy that a PC produces.

The big downside of using a case is that it takes up a lot of space. A typical mid-tower case is about 14x16x7 inches. A standard pin cab is 20.5" wide on the inside, and you'll have about 9 or 10 inches of vertical clearance between the floor and the playfield TV. That leaves enough room for a tower case lying on its side, but just barely.

Note that there's such a thing as a "small form factor" case. These take up less space, as the name suggests, but they're not really a good option for pin cabs. The big problem is that they don't accommodate full-size PCIe video cards. The video card is so critical to performance that you won't want to be limited to the few available small form factor options.

If you skip the case, it's straightforward to mount the motherboard, disk, and power supply to the floor of the cabinet. The only real complication is that the video card and other add-in cards will need some kind of structural support to keep them from working loose from the motherboard slots. One option is a partial case, known as an "I/O panel" or "I/O tray". You can search for these on the Web by looking for terms like "ATX I/O panel" or "mATX motherboard tray" (use "ATX" or "mATX" according to your motherboard's "form factor" spec). Another option is to fashion your own ad hoc support from wood or sheet metal. We'll look at specifics in Chapter 27, *Installing the PC*.

Fans

PC components and TVs generate heat when running, so you'll want to make sure the cabinet interior is well ventilated. Most cab builders do this by installing a couple of PC case fans in specially cut openings in the floor or back wall of the cabinet. See Chapter 28, *Cooling Fans* for more.

Network

You'll definitely want network connectivity in your cab PC, so that you can download software and pinball tables from the Internet.

Nearly all motherboards have built-in Ethernet ports for wired connections. If you'll have access to a wired router port in your pin cab's ultimate location, the built-in Ethernet port is all you'll need. If not, you'll want to consider another option, such as WiFi or powerline networking.

If you're already using WiFi for your other devices, you can get your pin cab on the network by adding a WiFi card. Some motherboards have built-in WiFi, so you might not even need to add anything; check your motherboard specs.

If you do add WiFi to your pin cab PC, I'd recommend doing so with a PCI add-in card that has an external antenna with at least a few feet of wire, to allow locating the antenna away from the motherboard. The reason is that the wood walls of a pin cab can substantially block the WiFi radio signal, so you'll get a much better signal if you can move the antenna outside of the cabinet. A built-in antenna or an antenna that's attached directly to the PCI card might not get a strong enough signal.

Another option is a "powerline" network. These send signals over your house's AC electrical wiring rather than by radio, so they're not susceptible to the interference and blockage problems that make WiFi problematic in some setups. They also don't require any extra wiring, since they use the existing power wiring in your house. To

make this work, you'll need one powerline adapter connected to the pin cab PC via the motherboard's Ethernet port, and a second powerline adapter connected to a port on your Ethernet router. Netgear, Linksys, and others make starter kits that come with the necessary equipment to set this up.

I always prefer a wired Ethernet connection when possible, since it's extremely reliable and almost effortless to set up. Powerline is my second choice when a wired Ethernet connection isn't possible, since it tends to be more reliable than WiFi and easier to set up. WiFi is great for mobile devices, but a pin cab has to be plugged into the power outlet anyway, so I think powerline is the way to go if you can't arrange a regular wired Ethernet connection.

Port connections

Assuming you're placing your PC inside the cabinet, you'll need a way to connect the keyboard, mouse, and (if you're using one) the Ethernet cable.

One easy way to deal with the keyboard and mouse is to buy wireless devices. Modern wireless keyboards and mice come with USB transceivers; just plug the transceivers into USB ports on the motherboard and you're set. Similarly, using WiFi or powerline Ethernet (see the Network section above) eliminates the need for external network cabling. Making everything wireless is the most convenient approach, but it's more expensive, and I've never been fully satisfied with the performance of any wireless keyboard or mouse I've used.

If you're using wired devices, a simple solution is to drill a hole in the cabinet big enough for the cables (preferably somewhere inconspicuous, like the floor or back wall), pull the cables through the opening, and plug them into the appropriate motherboard ports. The downsides of this approach are that it uses up a couple of feet of cable inside the cab (which might put your keyboard and mouse on too short a leash), and that it's inconvenient to disconnect and reconnect the devices (you have to open up the machine to get to the plugs).

If you want something a little more elegant and flexible, you can install the appropriate port connectors on the exterior of your cabinet and wire them to the motherboard internally. You can set this up pretty easily with parts made for installing data jacks in wall plates that resemble regular electrical outlet plates. These are commonly used for home theater and office installations. Here are the parts I'd recommend:

- 1 Keystone wall plate insert with 2 openings, for the keyboard and mouse
- 1 Keystone wall plate insert with 1 opening, for the Ethernet port
- 2 Keystone snap-in USB 3.0 female-to-female couplers
- 1 Keystone PS2 (6-pin mini DIN) female-to-female coupler (optional, if you're using an older keyboard with a PS2 connector instead of USB)
- 1 Keystone snap-in RJ45 Cat6 female-to-female coupler
- 4-foot standard male-to-male cables for each of the above connections

See Chapter 27, Installing the PC for installation instructions.

Optical disks

Most pin cab builders don't include any optical disks (CD-ROM or DVD-ROM) in their systems. And it almost goes without saying that floppy disks and other removable media are obsolete and can be skipped.

Apart from the operating system, you should be able to load all necessary software

by network download. The operating system itself can be installed from a USB thumb drive. Newer versions of Windows can be purchased on a pre-loaded thumb drive, and you can also use your existing desktop PC to create an installable thumb drive image from Windows DVD-ROM install media.

11. Power Switching

When you push the ON button your virtual cab, you want everything to turn on automatically: all the TVs, the audio system, the feedback devices, etc. Likewise, when you're done playing, you don't want to run around shutting everything off separately; you just want to press the OFF button and have the whole thing shut down.

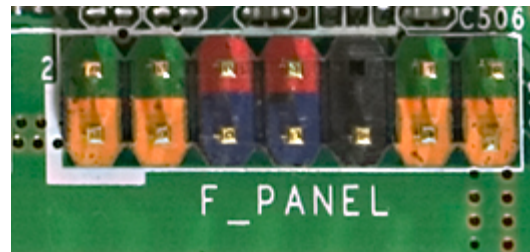
There are some challenges to achieving this kind of power integration, but they can all be overcome with a little planning and setup work. This chapter covers what you need to know to achieve single-button on/off control.

Soft power control through the computer

The key to whole-system power control is to let the computer control the power. Modern PCs are designed for "soft power" control, which means that the operating system software controls the power to the motherboard. This is how Windows shuts off power when you select "Shut Down" from the Start menu.

Using the soft power control on the PC motherboard itself is easy. You just need to wire a pushbutton to the power control pins on your motherboard. These pins are usually part of the "Front Panel" or "F_PANEL" header. If you were going to install the motherboard in a regular desktop case, the case would have a 10-pin connector that you'd plug into this header on the motherboard.

The exact location of the front panel header varies by motherboard, but it's usually easy to find. Look for a 10-pin connector near one of the edges of the motherboard. It's usually labeled F_PANEL or FRONT PANEL.



Intel defined a standard arrangement for this header a long time ago, so almost all motherboards use the same setup. You should check your motherboard's documentation to be sure, but the power switch pins are almost always the **red pins** in the diagram above - pins number 6 and 8 in the standard numbering.

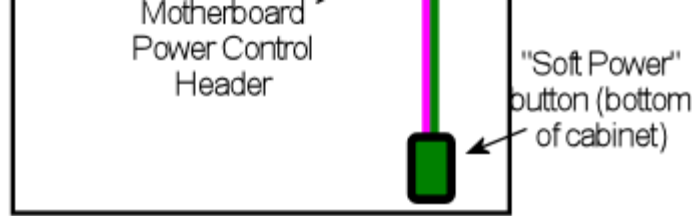
To power up the PC via the soft power control, all you have to do is connect those two red pins together for a moment. To turn the power off through Windows, just momentarily connect them again.

To connect a pushbutton to the soft power controls, you simply connect one terminal of the pushbutton to one of the red pins, and the other terminal to the other red pin. It doesn't matter which order they're connected in.

The standard place on a virtual cab to mount the power button is on the bottom of the cabinet, at the lower right corner. See the "Floor" section in Chapter 21, Cabinet Body for the standard location in the WPC cabinet plans.

Type of connector use with F_PANEL

The standard F_PANEL connector uses two rows of pins with 0.1" pin spacing. This is



Conceptual outline of how the PC can control power to other devices. The PC is plugged into an outlet that always receives AC power, allowing it to be switched on and off through the PC's "soft power" button. The TVs, audio system, and feedback devices are plugged into a second set of outlets that are controlled by an automatic switching control. The automatic control cuts AC power to the secondary outlets whenever the PC is off, forcing all of the other devices to turn off.

The final piece of the puzzle is how the outlet switch is controlled. It's not a "switch" in the sense of a wall switch that you operate manually. Rather, it's an electronically controlled switch, controlled by the PC's power state. When the PC is on, the switch turns on automatically. When the PC is off, the switch turns off automatically.

Since everything except the PC is plugged into switched outlets, all of those devices have no choice but to follow the PC's lead. When the PC is off, the switch cuts their power off at the source, so they have to turn off immediately whether they want to or not. They don't need any in-built circuitry to know whether the PC is on or off; we override all of that and control their power at the source.

Let's look at the two main ways we can implement this "switched outlet" setup.

Option 1: Smart power strip

You can buy pre-packaged "smart power strips" from Amazon or Best Buy that implement the sort of behavior we've been describing. This is supposed to be the easy way to implement a switched outlet, since it's plug-and-play. Just plug your computer into the designated outlet and the strip does its magic.

The reason I hedged by saying it's *supposed to be* the easy way is that it doesn't always turn out to be that easy. Some people run into snags with it, which we'll come to in a moment.

If you want to buy a smart power strip, try searching online stores for "smart power strip" and "green power strip". ("Green" because they save energy by cutting power to idle devices.) The specific product I used in my cab is an APC P7GB, which works well for me.

A retail smart power strip will have a specially designated "master" outlet or "computer" outlet, which is where you plug in the PC. The smart switching feature works by monitoring this special outlet to see if any power is flowing through it. When the sensor detects power flowing through the master outlet, the strip figures that the PC is on, so it turns on power to the other outlets. When the master outlet isn't drawing any power, the strip assumes that the PC is off, so it cuts power to the other outlets.

The good thing about this design is that doesn't require any special cooperation with the PC. It doesn't need any special connections to the PC or any special software. It works purely by monitoring the PC's power usage through its main power plug.

The weakness of the design is that nearly all PCs draw a little bit of power even when they're off, so the sensor in the smart strip that detects power usage in the master outlet has to be calibrated to allow for that. The smart strip can't just wait for the power level to drop to absolute zero, because that never actually happens. To make

matters worse, there's no "standard" idle power level; every PC is a little different, and it can even depend upon what's connected to the computer, since some USB devices draw power through the PC even when the PC is off. The maker of a smart strip doesn't know what kind of PC you're going to use with it, so they can't tailor the threshold level to your particular model. They just pick a level based on averages across many models. This *usually* works, but not always. Some PCs are so energy-efficient that they always stay below the threshold levels, so a smart strip might never detect that those PCs are on. Other PCs draw enough power when turned off that they remain above the threshold levels at all times, so a strip might never detect that those PCs are off.

Every strip has its own power threshold level, and every PC has its own power characteristics, so it's not easy to predict if a given strip will work with a given PC. The only way I know to find out is to test the specific pairing. So if you're going to test a smart strip, buy it from a store with a good return policy, in case it doesn't work with your computer.

Another disadvantage of the packaged smart strips is that they usually have a mix of switched and non-switched outlets, which means that they don't have very many switched outlets. For example, my APC P7GB only has three switched outlets out of 7 total, which isn't enough for all of the things I need to plug into switched outlets. That's easily solved by plugging a regular dumb power strip into one of the switched outlets to create more switched outlets, but that takes up extra space in the cab, so it would be tidier if the smart strip had more switched outlets built-in.

Option 2: DIY switched outlets

Note: there's a retail product called the IoT Power Relay that's almost exactly like the DIY solution we're about to describe, but it comes pre-built, saving you the work of finding the component parts and assembling them. You might also prefer it for safety reasons, if you're uncomfortable working with high-voltage wiring. See Option 3 below for more details.

A second way to implement automatic power switching is to build it yourself. This is more complex than buying a retail smart power strip, but it's more reliable and more flexible. It eliminates the problem that some smart strips have with properly sensing the on/off status of the computer. If you have any problems getting a smart strip to work with your computer, you can use this approach instead. This approach also makes it easier to add more switched outlets; the smart strips usually only have three or four switched outlets, which might not be enough for a decked-out pin cab. (With a smart strip, you can always plug a dumb power strip into one of the switched outlets add more switched outlets, but that takes up more space in the cabinet. If you build your own DIY switcher, you can start with a dumb strip that already has enough outlets for your needs.)

You'll need three things to build your own switched outlets:

- A small power strip (the ordinary "dumb" kind) with 2 or 3 outlets, to provide the **unswitched** outlets for the PC and the switched power strip
- A second ordinary power strip, with 6 or so outlets, to provide the **switched** outlets
- A 12VDC relay that can switch large power loads of at least 120VAC and 20A

For both power strips, I recommend buying strips equipped with surge suppressors. The primary strip will be running your PC, and the secondary strips will be running your TVs, so both would benefit from surge suppression.

Relays that switch large loads are also known as **contactors**. You can find suitable

devices on eBay, built into little circuit boards that simplify the wiring. Here's a picture of what to look for:

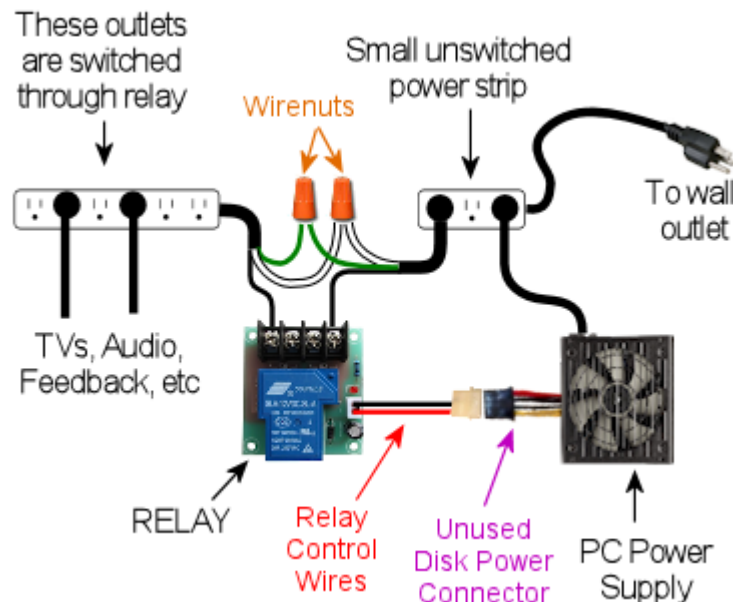


Search on eBay for "12V contactor board". You should be able to find listings that look similar to the picture above. (You don't need to find an exact match - the picture is just to give you an idea of what they look like.) The most common type currently listed has output limits of 250VAC and 30A, which is safely above our minimums. Make sure the control signal is listed as exactly 12VDC.



Be sure that your relay or contactor has a diode installed across the coil. This is important because it protects the 12V power supply and your PC electronics from the voltage spikes caused by the relay's magnetic coil. If you use an eBay contactor board, it'll probably have such a diode pre-installed, but you should visually inspect the board to make sure. If you're using a plain relay or contactor you bought as a separate component, you'll have to install a diode yourself. See Chapter 53, Coil Diodes for wiring instructions.

Here's the basic wiring diagram:



The theory of operation is simple. When the PC is ON, the PC power supply sends power to the disk connectors. This provides 12V to the relay, which turns the relay on, which in turn connects AC power to the switched power strip. When the PC is in one of the "soft off" modes, the PC power supply turns off power to the disk connectors, which cuts the 12V power to the relay. This switches the relay off, which cuts AC power to the switched power strip. This is equivalent to unplugging everything connected to the switched power strip, so all of the TVs and other devices will turn off.

Most of the connections shown are just a matter of plugging in power cords: plug the PC power supply into the unswitched outlet, plug the TVs and other devices into the switched outlet. But there are three DIY steps required:

Step one: Find an unused disk connector on your PC power supply. Connect the **control wires** from the relay board to the disk connector wires as follows:

- **Red** relay wire → **Yellow** power supply wire
- **Black** relay wire → **Black** power supply wire

See Chapter 45, How to connect 5V and 12V devices in the Power Supplies chapter for instructions on how to connect wires to the disk connector plug.

Step two: Make absolutely sure everything is unplugged for this step, because we have to cut into the AC power wiring.

On this step, we're going to cut the power cord in half for your **second** power strip: the one with 6+ outlets that's going to become the switched power strip. You don't have to cut it *exactly* in half, though; you should cut it where it will be most convenient for your physical layout. To figure out where that is, you should take a moment to do a rough fit in your cabinet to determine where you're going to situate the two power strips and the relay. Look at the diagram above and observe how the power cord from the second strip is going to be split into two parts, with the relay in the middle. Find a good point to cut the power cord so that you'll have a little slack on both sides of the cut when all of this is assembled.

Inside the power cord, you're going to find three internal wires. They should be color coded black, white, and green. The black wire is the one that we're going to connect to the relay. This is the "hot" or "line" wire that carries the voltage, so it's the one we want to interrupt to switch the outlets off.

The white and green wires are going to simply connect directly across both halves of the split power cord. In the diagram, we showed them connected by wire nuts, because we're assuming that you're going to have to cut the cord in half all the way through, severing all three wires inside. If you're really careful, you might be able to save that step by cutting only the black wire in half and leaving the white and green wires intact. If you can manage that, there's no need for the wire nuts. If you do end up having to cut the cord fully in half, though, reconnect the white and green wires by stripping a bit of insulation off the ends (about 1/2" worth), feeding the ends into a wire nut, and twisting them together until they're securely in place. Make sure there's no exposed bare wire sticking out of the nut when you're done. As shown in the diagram, connect green to green and white to white - all we're doing here is undoing the cut and restoring the green and white wires to their original condition. You might want to wrap the nuts and some of the surrounding wire in electrician's tape when you're done to secure everything in place.

The black wires connect to the input and output terminals on the relay. It doesn't matter which black wire goes to the input and which goes to the output; either way is equivalent. The relay terminals might be labeled **input** and **output** or **K0** and **K1**. Many of these boards have four terminals; when they do, each pair of terminals is simply connected together. For example, there might be two terminals labeled K0; these are wired together inside the board, so you can just pick one of the two to connect one black wire.

Step three: Secure everything in place and cover the high-voltage wiring for safety.

Once everything is wired, permanently fasten the relay board to the cabinet floor (or wall) with screws. I'd also recommend using standoffs, to leave a little open air under the board. Secure the power strips in place. I'd also secure the cut power cord

portions, perhaps with wiring staples, to ensure that the wire nut joints aren't jostled or stressed and that the black wires can't be accidentally pulled out of the relay terminals.

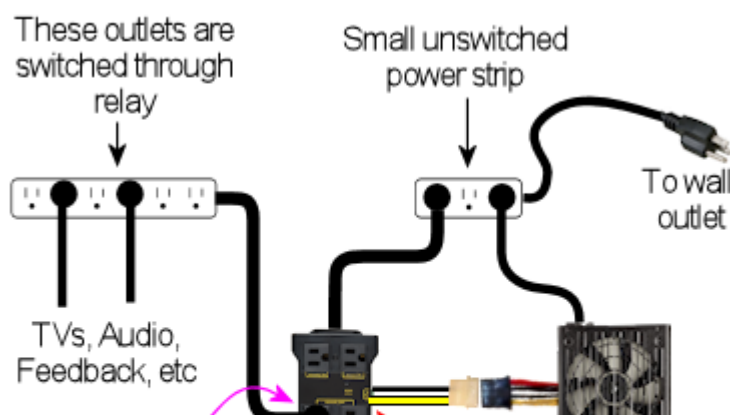
Finally, you'll have to improvise a cover for the entire relay assembly, so that there's absolutely no exposed metal or wire. The black wires will carry AC line voltage, which is hazardous high voltage. You don't want to allow anything loose in the cabinet to come into contact with the AC wiring, and you don't want any risk of touching it yourself while working in the cabinet. Remember that **the AC line voltage will be live on these wires whenever the cabinet is plugged in**, even when the computer is turned off. I'd recommend going to Home Depot and getting a plastic electrical junction box, of the type used inside the wall in your house wiring for switches and outlets. Get a box big enough that the relay board will entirely fit into it. Place it over the relay board and screw it into the cabinet so that the relay is permanently covered.

Option 3: IoT Power Relay

There's a retail product, called the IoT Power Relay, that implements the functionality described in the DIY option above, but without the need for you to buy individual components and assemble them. You can buy these from Amazon and other online retailers; search for **IoT Power Relay**. As of this writing (February 2021) they sell for about \$27.

The IoT Power Relay is set up to trigger based on just about any AC or DC voltage, so you can set it up exactly as described above for the DIY option, using the 12V wires (yellow and black) from one of your primary PC power supply's unused disk connectors as the trigger source. Note that the IoT Relay's trigger input is polarized, so you have to connect yellow and black in the correct order. Be sure that the yellow wire from the disk plug connects to the "+" terminal of the IoT Relay trigger input, and the black wire from the disk plug connects to the "-" terminal of the relay trigger input.

Once you have that wired up, just get an ordinary "dumb" power strip, and plug it into one of the IoT Relay's "Normally Off" outlets. The Relay may have outlets marked "Normally Off", "Normally On", and/or "Unswitched", depending on which revision you get. For our purposes, you can ignore everything except the "Normally Off" outlets. Those are the ones that switch ON when the trigger voltage from the main power supply switches on. Note that you don't even need an extra dumb power strip if you only need two switched outlets, since all versions of the IoT Relay have at least two switched outlets built in. For most cabs, though, that's probably not enough - you'll probably have four or five things that you want to plug into the switched power strip (secondary ATX power supply, 24V power supply, DMD or DMD video panel, backglass TV, audio amplifier). The extra power strip is just there to provide those additional outlets.





The TV Power Memory Problem

Now we come to the eternal bane of pin cab builders everywhere: power memory, or more typically, power forgetfulness.

If you've been following along for the first part of this chapter, your cabinet is now set up (or you at least have a plan) so that everything in it will turn on and off automatically with the computer. This happens thanks to our "smart strip", which controls AC power to every outlet (apart from the computer's own outlet) according to computer's power status.

The "power memory problem" in a nutshell is that many TVs won't turn on with this setup. Instead, they'll go into "standby" mode, where they'll stay dark while awaiting an IR remote control command. A TV in standby mode won't show a picture even if it's receiving an active video signal. This is bad for our "smart strip" system, because the smart strip makes the TV *think* it's being plugged in anew each time the PC is powered up. If the TV is designed to go into standby mode each time it's plugged in, the TV will effectively remain off, defeating our wonderful one-button power control.

How to tell if your TV has the problem

The only reliable way to determine if a particular TV has the power memory problem is to test it. If you're still shopping and want to test a TV before you buy it, you really have to find the exact model you're considering in a showroom or friend's house and test that specific TV. Don't count on similar models from the same manufacturer working the same way; it's not consistent across product lines.

The thing that really makes it hard to shop for this feature is that it's almost impossible to find good information about this online. You won't find it listed in a spec sheet or Amazon product page, and most people won't even know what you're talking about it if you ask. Your best bet is to ask on the virtual pin cab forums, because at least some people there will understand the question; even so, there are so many TV models that it's always hard to find someone who owns the exact one you're considering.

If you do have a way to test a model in person (or by proxy), you can get a definitive answer using the following text procedure. Ideally, you should try this using the same video input on the TV that you're going to use when it's installed in your cabinet. For example, if you're going to connect it to your PC by HDMI, run the test with the TV set to view an HDMI video source. The reason this is important is that some TVs have different behavior on this test with different sources.

Here's the test:

- Plug in the TV
- Turn it on
- Let it run for a couple of minutes
- Unplug the TV **without** turning it off first
- Wait a few minutes
- Plug it back in

On that last step, if it turns back on and returns to showing the same video source as before, hooray! The TV has good power state memory. It should just work automatically with a smart strip in a pin cab, so you shouldn't need to pursue any of the solutions below.

If the TV goes into standby mode after being plugged back in, it has the problem. You'll need one of the solutions below if you want to use it in your cab and you want single-button power control to work properly.

Solutions to the TV power-on problem

Fortunately, the power memory problem can be solved. Here are several possible solutions, in order of DIY-ness.

Solution 1: Buy a TV that doesn't have the problem

The easiest solution to this problem is to not have it in the first place. You can simply decide when buying a TV that power memory is a must-have feature, and reject any models that lack it.

My guess is that about 50% of the people in the pin cab forums would agree with that approach, because they really don't want to mess with any of the workarounds. Personally, I **don't** like this approach, because power memory is hardly the most important thing to me about choosing a TV. I think it's much more important to consider picture quality, motion blur, input latency, physical fit for the cabinet, price, and probably a few other features, before worrying about whether it has power memory. You might rule out some otherwise superior candidates if you consider this a deal-breaker. I'd only consider power memory a "nice-to-have" feature, meaning I'd only use it to decide between sets that are otherwise equals. The power memory problem is solvable by the other means we'll see below, so it's really not the end of the world if your TV needs a little help powering on.

Solution 2: Keep the remote handy

Of the 50% of cab builders who *don't* think power memory is the king of all TV features, I'd guess that about 50% of them throw in the towel on single-button power-up if their TVs don't have it. Because there's always the easy manual solution: keep the remote handy and press the On button every time you power up the cabinet.

This really isn't a terrible solution. I'm too much of a perfectionist to accept it for my own cab. It's not the inconvenience of it that's the problem for me; it's just that it makes the project feel a little unfinished. But in practical terms, it costs no significant amount of time and is only a minor inconvenience. If you can live with the rough edge, and the solutions below seem like more trouble than they're worth, you can stop here and call it done.

Solution 3: Tape down the On button

For some TVs, you can get away with a simple hack. It's inelegant (which is, after all, the proper definition of "hack"), and it doesn't work at all on most TVs. But it's worth trying, because if it does happen to work on your TV, it's a really simple solution that you can implement in a matter of minutes.

Here's the idea. On some TVs, if you keep the on/off button pressed down *all the time*, the TV will turn on and stay on whenever you plug it in. If your TV works this way, you can improvise some simple mechanical way of keeping the button pressed down permanently.

Before you start thinking about how to stick the button down, test your TV to see if

the trick works for it:

- Unplug the TV.
- Manually hold down the On/Off button.
- Keep holding down the button while you plug in the TV.
- Keep holding it down continuously for a couple of minutes.

Don't let go even briefly on that last step. The point is to test to see if holding the power button down for 10 seconds or 30 seconds or 60 seconds activates some special hidden action, like powering the TV back off, or rebooting it, or bringing up a service menu. "Long press" gestures often do something special like that on modern electronics, since everything these days needs to have a way to reboot it in case of software crashes. 30 seconds is almost always enough for a "long press" to take effect, but I'd give it a couple of minutes just to be sure.

If the TV turned on and stayed on, **and** you didn't activate some special hidden action by holding down the button for a long time, the hack will work.

To implement the hack, you just need to fashion something mechanical to hold down the button permanently. For some models, it's as easy as wrapping some duct tape around the bezel to apply pressure to the button. If that doesn't work for your TV's geometry, try taping a small object (a few pennies, perhaps) between the button and the tape, or try fashioning the right shape out of a paper clip or a little strip of sheet metal. If you have a 3D printer, maybe you can come up with the right shape for a custom plastic clip.

The big limitation of this hack is that it only works for certain TVs. Many TVs will respond by cycling repeatedly between On and Off or activating some special action. That's why you should try the test before worrying about how to implement the hack.

Solution 4: Pinscape TV ON system

If you're using the Pinscape expansion boards, there's a feature built in to help deal with TVs that won't turn on automatically when plugged in. The Pinscape boards have a power sensor that tracks the power supply status, and two mechanisms for sending an ON command to the TV: a relay that can be hard-wired to the TV's On/Off button, and an IR emitter that can be programmed to send the TV's IR remote control command code to turn on. These features can be configured in the Pinscape software to send the TV ON signal (by relay and/or remote) after an adjustable delay interval after the rest of the system powers up, to give the TV a chance to "boot up" and make itself ready to receive commands.

See Chapter 114, TV ON Switch for full details.

Solution 5: eBay timer board

You can build your own equivalent of the Pinscape TV ON feature using a type of electronic timer circuit board available on eBay.



Note: I recommend against using this solution, because it requires taking the TV apart; it's only included here for reference. If possible, use the Pinscape IR transmitter solution instead. See Chapter 114, TV ON Switch. The IR approach is non-invasive and fairly easy to build. You can use it even if you're not using Pinscape for anything else.

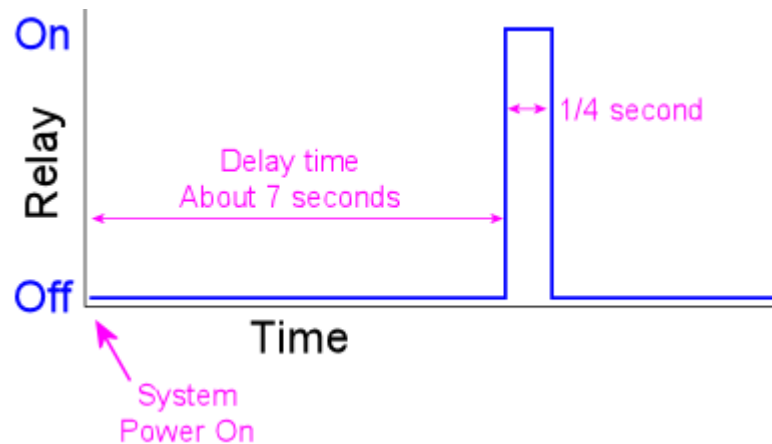
This approach works by simulating a manual button press on the TV's On/Off button, shortly after the system power is turned on. We don't physically press the button, but rather simulate it electronically, by soldering wires to the button's switch contacts

and connecting them briefly at the proper moment.

There are three important details required to make this work properly:

- We have to sense when the TV power switches from OFF to ON
- We have to wait a few seconds after that, to give the TV time to initialize
- We have to simulate a *momentary* button press only; we can't continuously hold down the button.

To accomplish all of this, we need a timer circuit. The circuit has to be triggered by the power coming on. It then has to pause for a delay time, long enough for the TV to get ready to accept command input, then it has "press the button" for just a moment. Here's what the timing looks like:



We're assuming that the timer is controlling a relay (an electronic switch). The "button press" is simulated by the relay toggling on briefly.

To implement this, we need the timer circuit itself, and then we need to connect it electrically to the TV's On/Off button.

Buying a timer: Suitable boards are available on eBay, but unfortunately it's rather difficult to find the needle in the haystack for this sort of item. The ones you're looking for are no-brand hobbyist products sold by Chinese companies, so there's not a particular store or product name I can point you to. You'll have to sift through the listings to find the right thing, but here's an eBay search term you can use as a starting point: "relay cycle timer".

To find the right timer, first make sure you find something with a relay. Most of the timer boards you'll find do use a relay, but some use solid-state switches (such as MOSFETs) instead. A relay is important for this application. Second, read through the descriptions and look for a list of "modes". The mode you're looking for should be described like this: "when the power turns on, the relay is disconnected, then delay T1, turn on the relay, delay T2, turn off the relay".

When you get the board, you'll have to program it according to the instructions (if any are provided) to set the correct mode and delay times. Set the initial delay time to about 7 seconds, and the second delay time to about 0.25 seconds. You can test that it's configured properly by cycling the power: each time you plug it into power, there should be about a 7 second delay, and the relay should click ON and immediately OFF.

Connecting to the TV: You'll have to be comfortable with taking the TV apart at this stage, because we have to connect some wires to the On/Off button.

There are no generic instructions for taking a TV case apart, so you're on your own for this part. Your goal is to open the case and expose the little circuit board containing the On/Off button.



Needless to say, use extreme caution with this step. In modern LCD TVs, the LCD panel and polarizing filter are very thin, brittle plastic sheets and often have no structural support other than the outer case, so it's very easy to crack them during the removal process or after the case is off. Removing the case will also void the warranty, so you're assuming the entire risk of breaking something by proceeding.

Once you get the case open, you should find a little circuit board located under the area where the buttons on the case are situated. It's usually long and narrow, and looks something like this:



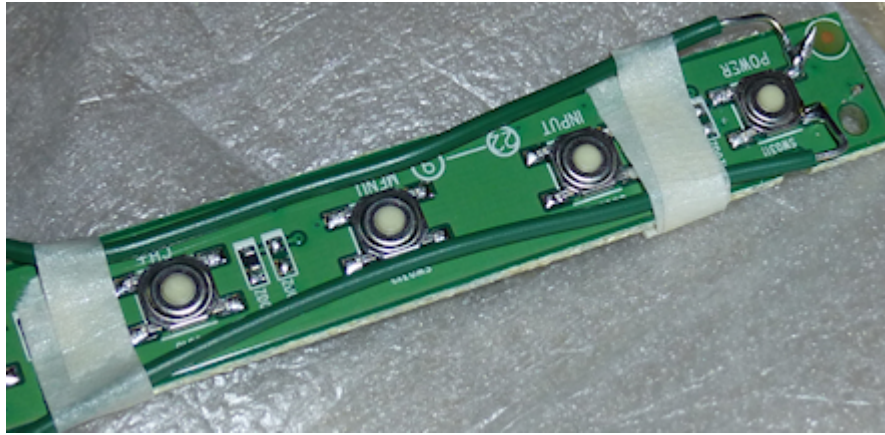
The red arrows in the photo above show the soldering points for the button leads. The little squarish silver objects are the buttons. These are normally situated immediately under the exterior plastic buttons on the TV's bezel; pressing on the exterior plastic button has the effect of pushing down on this metal part, which is the real button.

Once you find this circuit board, identify which button corresponds to the On/Off button on the outer case. Do this by position: just find the inner button that's situated underneath the On/Off button on the case. You can also do this by counting buttons from right to left, since there should be the same number of silver buttons on the circuit board as plastic buttons on the case.

Next, identify the switch leads. There are probably four leads to these switches, one at each corner. On the TVs I've looked at, the leads are in pairs that are electrically connected together, so there are really only two wires here even though it looks like four. Put your multimeter in continuity test mode and check the leads in pairs. Find a pair that are **not** connected normally, but that become connected when you press the button. These are the leads you want to solder to.

The next step is possibly even more delicate and tricky than opening the case. You have to solder wires to the button leads you just identified. To do this, use fine hookup wire, 24 AWG or thinner. Strip a very short length of insulation from the ends, around 1/8". Melt a little solder onto the end of the wire. Position the end of the wire at the desired contact point. Now get out some tape (I used thin strips of masking tape here) and secure the wire to the board a couple of inches away from the contact point. The idea is to hold it in place at the desired position before soldering so that the solder can just flow over the junction with everything already positioned properly. Once everything is in place, heat the end of the wire for a few moments, long enough for the solder to melt and flow onto the switch lead. Remove the soldering iron carefully and try to hold everything very still for a few moments so that the solder can solidify over the junction point. If all went well, the wire should stick to the switch lead. The connection will be delicate at best, so you'll want to secure the wire with a couple more pieces of tape to minimize mechanical stress on

it.



TV On/Off switch with wires soldered to leads

Repeat this process for the second lead. Once both are soldered and held securely in place with tape, test your work with the multimeter. Use continuity test again. Connect the meter leads to the free ends of the wires you just soldered. The meter should read open/no connection. Press the button, and the meter should read closed/connected. If that works, you're set. Put the TV case back together, taking care to run your newly attached wires out a suitable opening.

Now you just need to connect the newly attached wires to the timer board relay. Attach the wires to the relay **common (COM)** and **normally open (NO)** terminals on the timer board. (If the relay only has two switch terminals, those are the two to use!)

Finally, to power the relay board itself, connect its DC+ and DC- terminals to the appropriate voltage inputs from the **secondary** ATX power supply. For example, if it requires 5V for power, connect its DC+ input to the red +5V wire on the secondary power supply, and connect its DC- input to the black 0V/Ground wire on the secondary power supply. See Chapter 45, Power Supplies for Feedback for advice on connecting wires to the power supply.

Note that you **must** use a secondary ATX power supply to power the timer board (*not* the main PC power supply), and the secondary power supply must be plugged into the **switched** power strip. That's key to the whole scheme, because the timer board has to be powered up at the same time as the TV in order for the countdown to start at the same time the TV receives power.

Solution 6: DIY timer circuit

This works much like the eBay timer board described above, except that it saves you the trouble of tracking down the right item on eBay. The tradeoff is that you have to assemble your own circuit board instead. But you don't have to design the circuit: you can just build it from my plans.



As with the eBay timer board, I recommend against using this solution, because it requires taking the TV apart. If possible, use the Pinscape IR transmitter solution instead. See Chapter 114, TV ON Switch.

You can download the schematic, in EAGLE and PDF format, along with and an EAGLE printed circuit board layout, here:

mjrnet.org/pinscape/downloads/TVOnTimer.zip

Beta test warning: I haven't built or tested this incarnation of the schematic linked

above, which is an EAGLE rendition of the original hand-drawn schematic I used to build the TV ON timer in my own cab. The circuit I built based on the hand-drawn original is well-tested (I've used it for several years without a hitch), but I could have made errors doing the EAGLE translation. I also haven't done a test run of the board design, although my experience has been that EAGLE PCB layouts work fine as long as the schematic is sound. If you're willing to be a beta-tester for these plans, please let me know how it goes!

Before ordering parts, check your TV's timing! If you need different timing, you will need to order different values for parts C8 and/or R10. These parts determine the initial delay time. The delay time can be calculated from these as:

$$1.1 \times R \times C$$

where R is in Ohms and C is in Farads. With the default values as shown in the schematic, the delay is $1.1 \times 2.2\text{M} \times 2.2\mu\text{F} = 5.3$ seconds. Before you order parts, test your TV to determine if it requires a longer delay time:

- Unplug it
- Wait a few minutes
- Plug it in
- Use a timer to wait for 4 seconds
- Press the On button

If the TV turns on, try the test a few more times to make sure the timing is reliable. If so, the default 5 second delay should work. If your TV ignores the first button press on some trials, it probably needs a longer delay time. Try the test again with longer wait times until you find the shortest reliable waiting period. I'd add a second or two to the result as a cushion. Now you can reverse the timing formula above to find new values R10 and/or C8. For example, if you need a delay of 7 seconds, you could keep the resistance value the same and calculate a capacitor value of 2.89uF. Round up to the next common size, which in this case is 3.3uF, which would make the actual wait time about 8 seconds.

Build the board: Assemble the circuit, following the schematic or using the printed circuit board (PCB) design provided in the plans. The circuit is complex enough that I'd recommend building it on the PCB rather than ad hoc. You can have the PCB manufactured by OSH Park for about \$12 for three copies of the board, or at any PCB maker of your choice. You'll have two copies of the board left over to give to friends or use on your next cab!

Install the TV wires: The next step is to open your TV and solder wires to its On/Off button. The procedure is described in the section on eBay timers above.

Connect the board: Once you have wires connected to the TV's On/Off button, connect the other ends of the wires to one of the "K1" relay switches, on the Normally Open side. If you're connecting directly to the relay, connect to pins **4 and 8** or pins **9 and 13**. The relay in the spec is double-pole, meaning that it can switch two separate televisions on at the same time. That's why you have your choice of which relay pins to connect. If you have a second TV that needs the same treatment, you can simply connect it to the other pair of pins. If you use the PCB design, connect the TV wires to JP12 pins **1 and 2** (labeled "TV1" on the board silkscreen) or pins **3 and 4** ("TV2").

Connect power to the board: Finally, connect the power inputs to your **secondary** ATX power supply. As with the eBay timer, the scheme is predicated on the timer getting its power through a source that's switched on at the same time as the TV,

because the power-on time is the start of the delay timer countdown. If you're building from the schematic, connect VCC to +5V (a red wire) from the ATX supply, and connect GND to ATX ground (a black wire). If you're using the PCB layout, connect an ATX red wire to the JP7 +5V (marked on the silkscreen), and connect a black wire to JP7 GND. See Chapter 45, Power Supplies for Feedback for advice on connecting wires to an ATX power supply.

This circuit design is designed for this single function, so there's no need to "program" it as with the eBay timers. All you have to do is plug it in and it should work.

Solution 7: Use a USB IR transmitter

I'm only going to provide an outline for this solution, because I haven't tried implementing it myself. You'll have to do a little product research to fill in the details.

You can buy a device for your PC that lets you plug an IR transmitter into a USB port. Software on the PC can then command the IR transmitter to send a signal. You can use one of these to send the ON command to your TV via IR remote during the Windows boot process.

I don't have any specific product recommendations, but your best bet might be to search for "winlirc transmitter" or "winlirc blaster". winlirc is open-source software that lets Windows send and receive IR commands, so a winlirc-compatible device with a transmitter should serve the function we need here.

Once you find a suitable device, install it on your PC and arrange the IR emitter so that it's within range of your TV's remote receiver. Now you just need to set up a script on the PC that sends your TV's ON command while Windows is booting. You should be able to do this by creating a .CMD file containing the command line sequence to send the IR command, then placing a shortcut to the .CMD file in your Start menu's Startup folder.

12. I/O Controllers

One of the big things that elevates the virtual cabinet experience above ordinary desktop computer pinball is the ability to use real pinball controls: flipper buttons, coin slots, a plunger, "nudging" by actually nudging the cabinet. An equally big enhancement is feedback devices that create tactile effects, lighting effects, and mechanical sound effects that aren't just coming from a speaker.

If you're new to virtual pinball, you might wonder how all of this is possible, since normal PCs don't have any provisions for connecting any of these unusual devices. There's no "flipper button" connector on a Dell desktop. The secret ingredient is something called an I/O Controller ("I/O" for "Input/Output"). These are special hardware devices that plug in to the standard PC ports (usually USB) and provide the special wiring needed to connect buttons, accelerometers, plunger sensors, solenoids, lights, and so on. They provide the physical bridge between the PC and the unique pinball hardware.

This chapter gets into the details of what these devices do, and offers some suggestions for what to buy. The subject can seem overwhelming at first, because there are lots of product options, and they all have different combinations of features and functions. We'll try to make it easier by breaking things down by function, and giving you a comprehensive list of the products available and which functions they offer. Towards the end of the chapter, you'll find a product/feature matrix that shows everything at a glance.

I/O controller functions

Let's start by looking at the main categories of functions that these devices can handle.

Button input: A device that lets you wire regular pinball buttons to the PC is called a "key encoder". These devices are pretty easy to set up. You just run a pair of wires to each button (flipper, Start, etc) and connect them to the encoder. The encoder attaches to the PC with a USB cable. When you press a connected button, the encoder emulates either a keyboard key press or a joystick button press. As far as the PC software is concerned, you're just typing on the keyboard or using a joystick.

Nudge input: This type of device uses an electronic accelerometer to sense the motion of the cabinet. Good accelerometers are sensitive enough and accurate enough to detect when you nudge the cabinet and to measure how hard each nudge is. The pinball software can use this information to apply a corresponding acceleration to the virtual ball - in proportion to the strength of the nudge, so you can get realistic reactions for soft nudges, hard nudges, and a continuum in between. Nudge devices usually connect to the PC via a USB cable and emulate joysticks, so a physical nudge looks to the PC software like a momentary deflection on a joystick handle. The strength of the nudge is indicated by the magnitude of the deflection, which is what allows the software to differentiate between soft nudges and hard nudges.

Plunger input: This is a very specialized type of input device, because it has to use some type of position sensor to track the motion of the plunger, and then translate the readings from that sensor into a format that the PC can understand. Plunger devices usually attach to the PC via a USB cable and emulate joystick input. This is the same way most nudge sensors work, so we need a way for the PC to tell the two apart. This is usually accomplished by using different "joystick axis" assignments for each device.

Feedback output: Output controllers let you connect feedback devices to the PC so that the software can control them. As with the other devices, these usually use USB connections to the PC. Unlike the various input controllers, which all emulate ordinary PC input devices (mainly keyboards and joysticks), output controllers all need special software on the PC. Fortunately, the required special software is already integrated with the main pinball player programs.

Available devices

Now let's look at the available devices. Some of the devices fall neatly into single categories, and others can perform multiple functions.

Pinscape Controller, running on just the KL25Z (no expansion boards). Open source software, DIY hardware; about \$15 for the main microcontroller board. Key encoder, plunger input, nudge input, feedback device control.

This is an open source project that can handle all of the I/O controller functions with a single device. The main hardware required is a KL25Z, which is a \$15 microcontroller that comes fully assembled and ready to use. By itself, the KL25Z can handle button input and nudging (it has an excellent built-in accelerometer). Plunger input and feedback device control require additional hardware that's described later in this Build Guide. The Pinscape software does just about everything the various single-function commercial devices do, with some added bells and whistles of its own. It includes fully assignable button inputs (using keyboard keys and/or joystick buttons), a "shift button" feature, LedWiz emulation for universal software compatibility, "night mode", high-precision nudge input, high-precision plunger input, and numerous other features. It's highly configurable via its setup program (which runs on Windows, and is free and open-source), and the firmware itself is open-source, so you're free to customize it if you need to do anything beyond what the configuration options allow, or if you want to add whole new features. The firmware includes built-in support for several types of plunger sensor technologies, so you have a choice of different plunger setups.

The standalone KL25Z can handle button, nudge, and plunger input with little more work than attaching wires. It gets a little more complicated if you want to use it with feedback devices, because it needs some additional electronics to do that, as explained in Chapter 49, Pinscape Outputs Setup (Standalone KL25Z).

Pinscape Controller with expansion boards. Open source software and hardware design; components cost about \$100 for a full build. Key encoder, plunger input, nudge input, feedback device control.

This is an extension of the basic Pinscape Controller project that adds a set of circuit boards, primarily to provide more feedback device outputs. The boards make it possible to control a much larger number of feedback devices than the KL25Z can control on its own. The boards also provide built-in handling for high-power devices, so that you can connect things like motors, solenoids, replay knockers, fans, and flashers without any additional booster circuits. The hardware design is open-source, so you can build everything yourself from components, which add up to about \$100 for a full-featured build. You can also opt to build only sections of the boards if you only need a subset of the features, which reduces the cost accordingly.

Zeb's Boards plunger kit. Commercial, about \$140 from zebsboards.com. Plunger input, nudge sensor, key encoder.

This kit comes with the control board and plunger sensor that attaches to a

standard pinball plunger (available separately for about \$30). In addition to plunger input, Zeb's kit also handles nudging via an on-board accelerometer, and provides key encoding for up to 20 buttons (with fixed key mappings). Zeb's plunger gets the best user reviews of the commercial plunger options. It uses a high-precision sensor for the plunger that provides realistic plunger motion in the pinball simulation.

VirtuaPin plunger kit. Commercial, \$140 to \$160 from virtuapin.net. Plunger input, nudge sensor, key encoder.

The VirtuaPin kit comes with a control board and plunger sensor, and optionally includes the physical plunger assembly. Like other commercial plunger kits, the VP kit is very easy to set up, with little assembly required beyond attaching the sensor to the plunger. The control board has an excellent on-board accelerometer for nudge sensing, and has wiring for up to 16 button inputs. Button inputs are hard-coded as joystick buttons and can't be assigned to keyboard keys. If you're picky about realism in the plunger, be aware that this kit uses an IR proximity sensor to detect plunger position, and these sensors have relatively poor distance resolution. Some users have reported that the plunger animation can be choppy.

i-Pac 2 and i-Pac 4. Commercial, \$39/\$59 from ultimarc.com. Key encoder.

The i-Pac devices are full-featured key encoders. Their target market is video game cabinet builders, but they work equally well for virtual pinball, since the needs are basically the same. Buttons are fully assignable (via a setup program on the PC) to keyboard keys and joystick buttons. The devices have a "shift button" feature that lets you assign two meanings to each physical button by holding down a designated shift button to activate the second meaning.

i-Pac Ultimate I/O. Commercial, \$99 from ultimarc.com. Key encoder, feedback device control.

This is a hybrid of the i-Pac and PacLED devices that provides button input encoding and feedback device control. The key encoder features are just like the i-Pac devices, with 48 button inputs. The feedback output controller is designed specifically for attaching 32 small (20mA) RGB LEDs. For a virtual pinball cabinet, you'll want to attach other devices that require higher power, so you'll need external booster circuitry, such as Zeb's booster board. One warning: as of this writing, this device's output controller feature isn't as well supported in the standard virtual pinball software as the LedWiz and PacLed devices, so you might encounter some difficulty setting up the software to take advantage of it. The button input feature will work seamlessly, though.

LedWiz. Commercial, \$45 from GroovyGameGear.com. Feedback device control.

The LedWiz was the first output controller widely adopted among virtual pinball cabinet builders, and as a result, it's the most universally supported option. This device is aimed at video game cabinet builders, so it was designed especially for controlling LEDs (thus the name), but it's not limited to LEDs. It can control just about any type of device. The caveat is that it has a low limit on how much current it can control per device (500mA), so you can't connect high-power devices directly. You can work around that by adding an external booster board to increase its power limits. That 500mA limit is adequate for most types of lights, including flasher LEDs and button lamps. A booster is needed for most mechanical devices, like knockers, motors, and solenoids.

PacLed-64. Commercial, \$59 from ultimarc.com. Feedback device control.

This device is well supported by the newer open-source pinball software systems (including Visual Pinball and PinballX), but it's not as compatible with older systems like Future Pinball as the LedWiz is. It provides 64 outputs for small LEDs. Like the LedWiz, this device was designed for video game cabinet builders, but its power handling is even more limited and isn't sufficient for high-powered lights like flashers and strobes. So you'll need to combine this with a booster board for almost anything in a virtual pinball cabinet.

SainSmart USB relay boards. Commercial, about \$20-\$40. Feedback device control.

SainSmart makes USB-controlled relay board with 8 relay outputs. Software on the PC can send USB commands to turn attached devices on and off through relay switches. The relays can be used to control devices that use high power levels, so they're good for devices like solenoids, contactors, and replay knockers. However, these boards aren't a good choice for lighting devices, since relays on simple on/off switches and thus can't control brightness. For lights (especially flashers and button lights), you'll want to be able to control the intensity level of each output. The other slight disadvantage of relays is that they add a small lag time for switching devices on and off, which can make the device response slightly out of sync with the game action. Most people don't find this noticeable, though.

Warning! DOF is currently only compatible with the **8-relay** Sainsmart boards. Sainsmart makes the boards in different sizes, from 4 to 16 channels, but DOF **only** works with the 8-relay version.

Warning! There seem to be some no-brand devices out there that look ridiculously similar to the Sainsmarts, with the same blue lays laid out the same way, but which aren't compatible at the software level. That means they won't work with the existing pinball software, unless you can do some additional programming to add support yourself. I'd avoid look-alike boards that aren't clearly branded as Sainsmart products.

Zeb's Boards booster board. Commercial, \$75 from zebsboards.com. Feedback device add-on.

This board lets boost the power from 16 outputs on an LedWiz or PacLed output controller. The booster board itself isn't an output controller, so you can't use it alone; it has to be used in conjunction with one of the output controllers. The booster board raises the power level on 16 of the output controller's ports to 6A, which is enough to control anything in a pin cab, including high-power devices like replay knockers, shaker motors, gear motors, fans, beacons, and solenoids. If you need more than 16 boosted ports, you can add more of these boards to boost an additional 16 ports per board.

SainSmart (non-USB) relay board. Commercial, \$20 to \$40. Feedback device add-on.

These boards are similar to the SainSmart USB relay boards, but they're not controlled by USB. Instead, they're controlled by individual inputs to the relays. You can connect the relay control inputs to the output ports of an LedWiz or PacLed unit to boost the power handling capability of the controller via the relays. You can then attach a high-power device, such as a replay knocker or solenoid, to the relay. The controller unit will switch the relay on and off, and the relay will in turn switch your high-power device on and off. This is a simple way to boost the power handling of an LedWiz or PacLed unit. Note that the relay switching adds a small amount of lag time, which can make the feedback

response slightly out of sync with the game action, although most people who have set these up don't find this to be noticeable.

Zeb's Boards output kits. Commercial, \$550 to \$900 from zebsboards.com. Feedback system including controller *and* feedback devices.

These kits offer turnkey feedback setups that include not only the output controller device but also all of the feedback devices themselves, all fully assembled and wired. Everything comes pre-mounted to a couple of modular panels for easy installation in a cabinet.

Recommendations

For the DIYer: I'm biased, obviously, but if you like building things yourself, my pick would be Pinscape. For a fully decked-out system with all the feedback devices, go with the expansion boards. For the input features only (buttons, plunger, nudging), the standalone KL25Z is all you need. I'm pretty sure Pinscape has all of the features of the best-of-breed commercial products (plus some extra features they don't have), equal or better performance, and a lower price tag. And the open-source design puts you in complete control. You can change anything that's not to your exact liking; and if you take "DIY" especially seriously, you can use my code as a starting point and rewrite as much of it as you want from scratch.

If you want "no compromises": Again, I'm biased, but I think the answer here is Pinscape. It has the most full-featured and highest performance implementation I'm aware of for each of the components. It's highly configurable through its Config Tool, so you can set it up exactly how you want it. And again, it's open-source, so if there is anything you want it to do that it doesn't already do, you can add it; or if there's anything it does do that's not quite the way you want it, you can change it.

If you're uncomfortable with DIY: You'll probably be happier with the commercial options if you're not comfortable building this sort of thing yourself. The commercial products come ready to install, with only some basic setup required. The big challenge is figuring out which devices you need, since their functions overlap in somewhat confusing ways. Here are my recommendations for some common scenarios:

For a simple feedback system with lights only: If the only feedback devices you want are lighting devices (flashers, strobes, and button lights, for example), I'd recommend an LedWiz as the output controller. The LedWiz is inexpensive, and for just lights, it's simple to set up, since that's exactly what it's designed for. A single LedWiz has plenty of ports for a pin cab's lighting needs. The LedWiz is a good choice for lighting devices because it can display a range of brightness levels, which allows for fades, flash patterns, and RGB color mixing effects. The LedWiz isn't as ideal for high-power devices like solenoids and motors, since it can only handle limited power to each port; while it's possible to use it for these devices, you need additional hardware add-ons, which largely negates the whole "it's simple" advantage.

For a simple feedback system with solenoids and motors only: If you want a feedback system consisting only of tactile effects (replay knocker, flipper and bumper solenoids, shaker motor), get a SainSmart USB relay board. I'd get the 16-output type so that you have plenty of outputs for extras you might want to add later. The SainSmart board is the easiest thing to set up for high-power devices. The downside is that relays are strictly On/Off switches, so the SainSmart can't display different brightness levels if you use it to control lights - it can only turn them fully on and fully off. That makes it good for devices like solenoids and

motors, but not so good for lamps and LEDs, where you need brightness control to get the full range of effects. The other disadvantage is that the relays are mechanical, so they can eventually wear out; some people on the forums have reported having to replace their SainSmart boards every couple of years due to relay failure.

For a plunger-less system: If you don't want to include a plunger in your setup, use a KL25Z running Pinscape as the input device. You don't need the expansion boards if you're just using the input features. The installation work for buttons and nudge input is pretty much the same as for any of the commercial options, and Pinscape is a lot cheaper and has more features.

For a turn-key plunger: If you want a plunger but don't want to build the electronics yourself, buy Zeb's plunger kit. It's easy to set up and gets generally good reviews from users.

For a turn-key feedback system: If you're the opposite of a DIYer, and you don't want to do a lot of planning or parts sourcing or assembly work, buy one of Zeb's pre-built feedback kits. They're expensive, but they'll save you a lot of work, and they'll eliminate any anxiety you might feel about the things going wrong if you build it yourself.

Feature matrix

Here's a summary of the key features of the available controllers, to help you decide on a combination of devices for your system based on the features you plan to include.

Device		Key Encoder			Plunger	Nudge		Feedback Output			
Name	Type/ Price	# Buttons	Assignable	Shift Button	Sensor Type	Precision	✓	# Outputs	Power Limit	Brightness Control	Booster Required
Pinscape (standalone)	Open source \$15	24+	✓	✓	Multiple options	High	✓	22+	4mA	✓	Yes
Pinscape w/ expansion boards	Open source ~\$100	24+	✓	✓	Multiple options	High	✓	65-128	4A ¹	✓	No
Zeb's Boards plunger kit	Commercial \$140	20	-	-	Poten- tiometer	High	✓				
VirtuaPin plunger kit	Commercial \$140	16	-	-	IR	Low	✓				
i-Pac 2	Commercial \$39	32	✓	✓							
i-Pac 4	Commercial \$65	56	✓	✓							
i-Pac Ultimate I/O	Commercial \$99	48	✓	✓				96	20mA 6@1A	✓	Yes ²
LedWiz	Commercial \$45							32	0.5A	✓	Yes ³
PacLed-64	Commercial \$59							64	20mA	✓	Yes ²
SainSmart USB relay board	Commercial \$20-\$40							4- 16	12A		No
Zeb's Boards booster board	Commercial \$75							16	6A	✓	No ⁴
SainSmart relay board (non-USB)	Commercial \$20-\$40							4- 16	12A		No ⁴

Zeb's Boards output kits	Commercial \$550- \$900								16- 64	1A- 6A	✓	No
--------------------------------	-------------------------------	--	--	--	--	--	--	--	-----------	-----------	---	----

Footnotes:

1. The 4 Amp limit applies to the general purpose outputs on the power board. There are 32 of these on each power board. In addition, the main board has 16 flasher/strobe outputs that can handle 1.5A each, and 16 outputs for button LEDs that can handle 20-50mA each. The typical setup uses one main board and one power board, which gives you 65 total outputs, plenty for a decked-out cab. If you need more, you can add extra power boards for another 32 of the high-power outputs per, up to the software limit of 128 total outputs.
2. This device's outputs are designed to drive low-power LEDs, which it can do without any extra booster circuitry. A booster board is needed to drive anything needing higher power, such as flasher LEDs or mechanical feedback devices.
3. The LedWiz can handle 500mA per output, which is sufficient for most types of lights, including LED flashers and button lamps. A booster board is required for most non-lighting devices, such as contactors, replay knockers, solenoids, fans, shakers, and gear motors.
4. This device works in conjunction with one of the output controllers (LedWiz, PacLed-64, etc). It can't be used alone; it has to be used in combination with an output controller.

13. PC Hardware Setup

If you bought an assembled PC, all you have to do at this stage is unpack it. If you're building a PC from parts, you might want to do a preliminary test build at this point to make sure everything's working, and to confirm that you have everything you need.

If you're going to install your PC equipment in your pin cab without a standard PC case, you can do your test build with the parts spread out on a tabletop. Be sure to do your work on a non-conductive surface like wood to avoid accidental shorts.

Static electricity precautions

Many of the parts in a PC are sensitive to static electricity, particularly the CPU and memory chips. That's why everything comes packaged in those silvery plastic anti-static bags.

Your body can accumulate a significant static charge, enough to damage semiconductors, so you have to be careful handling these parts to avoid zapping them when you touch them.

The way to protect against damage when handling static-sensitive parts is to frequently "ground" yourself, meaning that you electrically connect yourself to the earth. Professional engineers do this with something called a grounding strap, which is a conductive bracelet that you wear on your wrist and connect by a wire to your house's ground wiring. That keeps you connected to the earth ground the whole time you're working. You probably don't have one of these unless you do a lot of electronics work, but you can achieve the same thing by simply touching a metal surface that's connected to earth ground periodically while you're working.

Where do you find a grounded metal surface to touch while working? When you're working on a PC, there's one that's always close at hand: the power supply case.

Here's what I suggest. Before you start any other work, get your power supply out of the box and **plug it in**. Put it on the table where you're going to do your work. Find a **bare metal surface** on the power supply case. If there are no bare metal surfaces, an unpainted screw head on the case will work. The key is bare metal attached to the case.

As long as the power supply is plugged in with a three-pronged plug, you can now ground yourself at any time by touching that bare metal surface.

You don't have to keep in contact with the grounded metal all the time, although it's ideal if you do (which is why they invented those grounding bracelets). But do ground yourself frequently while working, at least every few minutes, and every time you return to the work station after walking around. Walking around is a great way to accumulate static charge. Another good time is whenever you're about to open an anti-static bag or start handling a new part.

Assembling the motherboard

You should find detailed installation instructions for assembling your motherboard in its packaging. If you bought a retail-packaged CPU, it should also include its own instructions for installing it in the motherboard.

You should follow the setup instructions in your motherboard's documentation, since every board is a little different. In general, though, here are the steps:

- Install the CPU in its socket. Be especially diligent about static electricity precautions while handling the bare CPU.
- Install the CPU fan. Most motherboards have clips or sockets for securing the fan, so this is usually just a matter of fitting the fan into place.
- Connect the CPU fan wiring. The CPU fan should have a short wire connector that mates to a "CPU FAN" socket or pin header on the motherboard.
- Plug in the memory (RAM) chips.
- Insert the video card into its motherboard slot. Note that some motherboards have multiple slots that are physically capable of holding the video card, but one slot might be better than the others because it has a faster data connection. Check your motherboard documentation to see if one slot is designated as the special slot for the video card. The documentation might not put it in these terms; it might instead list the PCI "x" speeds for the different slots, such as x1, x4, or x16. The highest "x" speed is the fastest slot, so it's the one to use for the video card.
- Connect the storage device (hard disk or SSD).
- Connect the power supply to the motherboard. There are usually two connectors from the power supply that plug directly into the motherboard.
- Connect the power supply to the video card. Most higher-end video cards have a dedicated connection directly to the power supply. Your power supply should have the special mating plug attached to one of the wires in the bundle of cables coming out of the supply.
- Connect the power supply to the hard disk or SSD. These devices have their own power connection. As with the video card, the hard disk/SSD has a dedicated power connector that will mate with one of the connector wires coming out of the power supply.
- Connect a video monitor to the video card.
- Connect a keyboard and mouse to USB ports on the motherboard.
- If you have a wired Ethernet network, connect a cable to the network plug on the motherboard.

Video card precautions

If you're assembling everything on a tabletop without a standard PC case, the video card's connection to the motherboard will be fragile, since it won't have the structural support that a normal case provides. The PCI slot is really only meant to provide the electrical connection; it's not meant to provide structural support or anchor the card in place.

You'll definitely need to secure the card physically in your eventual installation in the pin cab. For testing purposes, though, you can work with this flimsy setup as long as you're careful not to nudge anything while the power is on. Even a little nudge, like someone bumping into the table, can be enough to momentarily interrupt the electrical connection in the socket. I'd avoid that; in most cases the only harm will be to make the PC reboot itself immediately, but these cards really aren't meant to be hot-plugged (taken in and out with the power on) and could be damaged by this.

Power it on

Once you have everything assembled as described above (and according to any additional instructions in your motherboard's documentation), you're ready to give it

a test run.

Modern PC motherboards have "soft power" controls, so even though it's already connected to the power supply, it won't actually turn on until you press the "on" button. If you have a case, the "on" button is the one on the case. If you're working without a case, though, you have to find the "Power Switch" pins on the motherboard. Check your motherboard documentation to find the right pins. The motherboard manual will tell you that these are the pins to connect to the "Power Switch" connector wires from your case.

Once you identify the "power switch" terminals, you turn the PC on simply by shorting these two pins together for a moment. They're usually right next to each other on the motherboard, so if you don't have the right kind of connector handy, you can simply touch a metal screwdriver tip to both pins at the same time. (Be careful not to touch any other pins while doing this, of course.)

BIOS Setup

When the PC first powers up, you should see a brief message flash on the screen telling you to press a key on the keyboard to enter the BIOS Setup. It's usually one of the function keys, often F8 or F12, but it varies by motherboard. You'll just have to watch for the message to find the right key. You should also be able to find this information in your motherboard's documentation.

You should be able to reboot by pressing Alt+Ctrl+Del on your keyboard, which should give you another chance to press the magic BIOS Setup key. You can also power cycle by shorting the "Power Switch" pins together again to turn the PC off, then wait a few seconds and do it again to power up again.

You have to press the magic BIOS Setup key at just the right moment after the power comes on. It usually works to tap on the key rapidly while the machine is powering up.

The BIOS Setup lets you configure the machine's hardware and verify that everything you physically attached (memory, disk, video card) is being properly recognized by the motherboard. It's worth running the setup as a very basic test, since the fact that it runs at all confirms that the video card, keyboard, CPU, and memory are all working.

14. Windows Setup

Once you have the PC hardware set up, the next step is to install Windows.

We won't try to provide a Windows installation tutorial here, since the Web has much more comprehensive information on that than we could provide here - and you probably won't need to look at any of that anyway, since Microsoft has managed to make the process fairly automatic in most cases. But we do have some recommendations for settings specifically for pin cabs.

So continue below after you've gone through the basic Windows installation procedure.

DON'T turn off UAC

User Account Control (UAC) is a Windows security feature added in Windows Vista, and present in all Windows versions since, that makes Windows ask for your permission when an application tries to do something that affects core resources in the operating system, such as altering a system registry setting.

You might see advice on the forums telling you to turn off UAC to avoid those prompts. I recommend ignoring that advice. **Leave UAC enabled.**

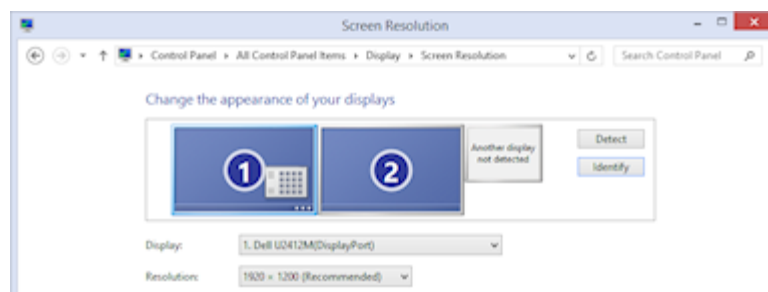
I recommend **against** disabling UAC because doing so can cause software compatibility problems. UAC is an integral part of modern Windows systems, and removing it actually changes the way Windows works internally, which can break some application software. If you're technically inclined and curious about the details, see Mark Russinovich's article in Microsoft's TechNet Magazine, June 2007, Inside Windows Vista User Account Control. For a less technical explanation, try a Google search for "Why not disable UAC". Disabling UAC also increases your vulnerability to system damage from malware and unintentional software bugs.

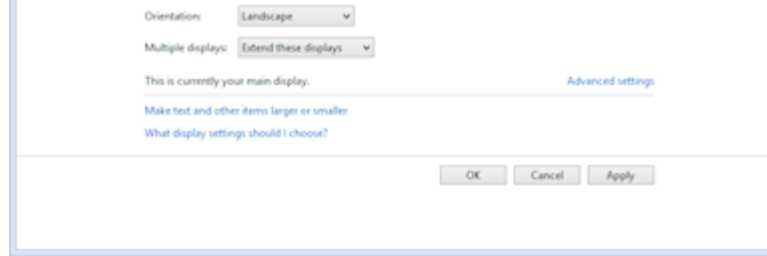
Any advice you see about disabling UAC is likely outdated. The notion comes from the early days of Windows Vista, when UAC was first introduced. A lot of software at the time wasn't properly designed for the tighter security rules added in Vista, so some people resorted to disabling UAC in an attempt to keep their old software running. As time has gone on, though, most software has been updated to work properly, and UAC itself has been improved to make it less intrusive.

Arrange monitors

If you're using multiple monitors, Windows will combine their display areas into a single large virtual desktop.

Windows lets you adjust the way the monitors are arranged within the virtual desktop via a control panel. The idea is that if you have two physical monitors sitting side by side on a desk, you can arrange the virtual desktop to match. To reach this control panel, bring up the Display control panel, then click Adjust Resolution.





This control panel shows a diagram of how the physical monitors are lined up across the virtual desktop. You can drag the monitors in the diagram to rearrange them. How you arrange your monitors is mostly up to you, but there's one important rule you should follow:

The main display should be at the upper left of the virtual display area.

Note that the "main display" isn't necessary display #1. The numbering is just a way to identify the monitors and is somewhat arbitrary. The "main display" is simply the one you designate as such using the "Make this my main display" button in the control panel.

I recommend the following layout:

- Make your playfield monitor the main display
- Arrange the monitors in a single row
- Make sure the main display is at the left end of the row

Some versions of Windows only allow certain monitors to be the main display, so you might not have the option to make your playfield monitor the main one. If you can't, you should still arrange things so that the main monitor is at the left end of the row.

The reason I recommend this arrangement is that some software, notably VPinMAME, can have odd problems if the main monitor isn't at the left extreme of the virtual desktop. Windows internally assigns the "origin" of the pixel coordinate space to the top left of the main monitor, so any monitor that's to the left of this (or above it) in the virtual desktop area will have negative pixel coordinates. Some software (like VPinMAME) gets confused by the negative coordinates. If you don't follow this advice about the layout, VPinMAME won't be able to properly remember your screen layout, because it incorrectly interprets negative coordinates as errors.

Turn off "Sticky Keys"

Most pinball software uses the Shift keys to control the flippers. Windows has an accessibility feature called "Sticky Keys" that locks the shift keys on if you press them several times in a row. The feature is well-intentioned - it's there to help people who have difficulty pressing several keys at once - but it interacts horribly with pinball games. It can make the flipper get stuck in the up position after a bunch of rapid flips.

Sticky Keys is an accessibility feature, so you'll find it on the "Ease of Access Center" control panel, which you can find in the main Control Panel window.

On most versions of Windows, you can also find this control panel by pressing Windows+S ("Search") and typing "Sticky Keys" into the box. Look for "Change how your keyboard works" or "Make the keyboard easier to use" in the search results.

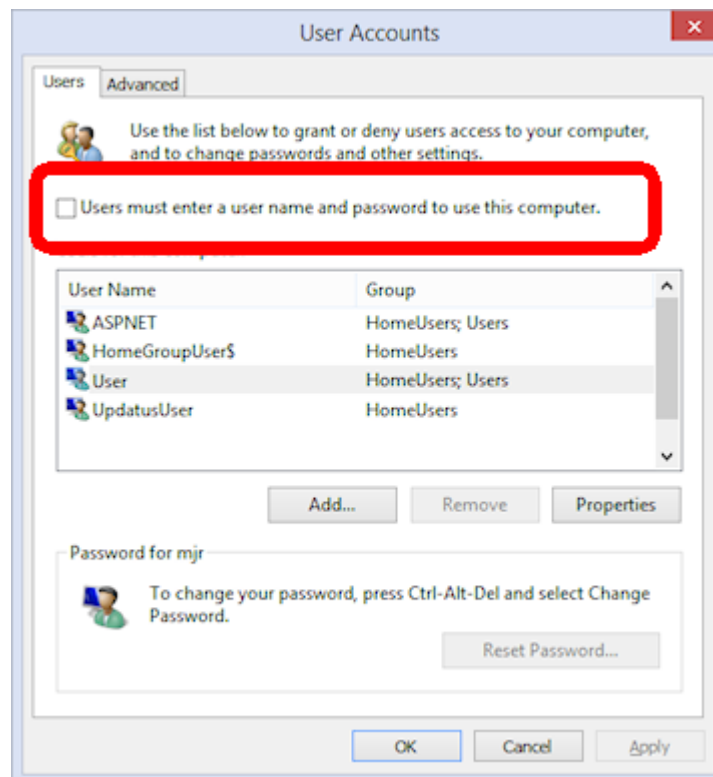
Once you find the dialog, look for the checkbox labeled "Turn on Sticky Keys". Make sure it's un-checked.

Automatic login

Windows normally asks you for a username and password every time you start up the computer. This is great for office or work PCs. It's not great for pin cabs, where you want to be able to turn on the machine and get straight to playing pinball. It's a little crazy to have to get out the keyboard and log in first.

Fortunately, Windows lets you disable the login requirement, so that the machine boots straight to the Windows desktop each time you turn it on. Here's the procedure:

- Press Windows+R ("Run Program")
- Type **netplwiz** into the Run box and press Enter. This should bring up the advanced user accounts control panel (titled "User Accounts" on most Windows versions).



- Look for the checkbox "Users must enter a user name and password to use this computer" (circled on the screen shot above). Un-check this box.
- Click OK to confirm the changes.
- A new dialog should appear asking you to enter the credentials to use to sign in automatically when Windows reboots. Enter the username and password you wish to use and click OK.

The next time you boot, Windows should automatically log in to the account you selected and go straight to the desktop.

Remove (or don't install) anti-virus software

On any gaming PC, it's best to minimize the number of background tasks running. Background tasks take CPU time away from the main program that's running. This can have a visible effect on the animation in a game, since even a very short interruption in the animation updates can cause momentary glitches and stutters.

Probably the most important background task to get rid of is third-party antivirus or anti-malware software. Virtually all of these programs use significant system resources and will noticeably hurt game performance. If you built the PC yourself and did a fresh install of Windows, you can simply elect not to install any third-party security software. If you bought a pre-built PC, and the vendor larded it up with "free trial" security software, I'd remove it all.

It might seem crazy in this day and age to run a PC without any security software, and I certainly wouldn't recommend going without on an ordinary PC, but a pin cab isn't an ordinary PC. The difference is that you'll probably only use it for playing pinball - not for browsing random Web sites, opening random emails, or downloading random programs. As long as you're careful about what you install, your risk of encountering any malware should be small. Stick to the well-known pinball programs and add-ons, and always get them from reputable sites.

An exception: you can (and should) leave the built-in Windows security features enabled, particularly Windows Defender and the Windows Firewall. Those have a negligible impact on system performance, and they provide a good baseline level of protection.

Backing up your system data

Everyone knows how important it is to back up the data on a PC, in case you ever need to recover from hardware failures, accidental file deletions, or malware attacks. It's a lot of work to set up all the software on a pin cab, so backups are as important for a pin cab as for any other PC.

The approach I've used for a long time is to back up to external USB hard disks. Those are reliable and fairly inexpensive, and most of them come bundled with backup software. More recently I've added cloud backup as a second layer of protection. There are several good on-line backup services that run about \$10/month for reasonable storage quotas.

Here are some things I consider important when setting up your backup plan:

- It should be **automatic**. It should run on a schedule so that you don't have to remember to run it yourself. It's too easy to put it off or forget about it entirely if you have to do it manually. The cloud backup services make this particularly easy.
- The media should be **offline** between backup sessions, meaning not physically connected computer you're backing up. This will protect your data in case of a hardware failure (such as a power spike that fries everything connected to the computer) or a system-wide malware infection. If you back up to an external USB disk, simply unplug it from the computer after each backup.
- Better still, the media should be **off-site**, at a physically separate location. This will protect your data in case of a whole-house disaster like a fire or flood. This is a big benefit of cloud services.
- Backups should be **versioned**. Versioning is particularly critical for malware protection, because an infection might not be immediately apparent, so your most recent backup might include infected files without your knowing it. Keeping multiple versions lets you go back in time to a point before the infection. Versioning is also a nice safety net in general - it lets you go back to an older working configuration if something goes wrong with a software update, for example.
- The backup software should do a whole-disk scan. If you have to manually

choose the files that get backed up, you'll inevitably miss something important. I always prefer starting with a default that includes everything on the disk, and then manually selecting files to exclude.

- The backup scan should **include the Windows registry** as part of the backup, since Windows itself and many application programs store a lot of important configuration data there.

15. Pinball Software Setup

Now let's look at the software needed to transform this from a plain Windows PC to a pin cab machine.

The main software you need, of course, is the pinball simulator. To take advantage of the special features of a cab, you also need some add-ons to display the backglass artwork and control the feedback devices. You'll also want a "front end" program that provides an interactive menu for selecting tables to play. We'll look at the options for all of these in this chapter.

Should I use "Run as Administrator" for everything?

The simple answer is No. You might find advice in the forums or FAQs or other guides saying that you should routinely run everything in Admin Mode. My advice is to ignore that other advice, because it's usually outdated or misinformed.

What is "Admin Mode" anyway? Microsoft divided things into "Admin" and "User" spaces to protect the internals of the system from being accidentally damaged by software bugs or user errors, or intentionally damaged by malware. Programs running in User Mode have some restrictions on what they can access, but Admin Mode programs can access everything. Admin Mode is supposed to be reserved for special system programs that have a legitimate reason to modify your system internals - programs like installers, disk management tools, and system control panels. Everything else is supposed to run in regular "User" mode.

So why do so some people on the forums tell you that you *should* use Admin Mode routinely, when Microsoft says you shouldn't? The reason is mostly "history". In the old days, there were a few isolated software components in the pinball simulation ecosystem that really did require Admin Mode to work properly. The snag is that Windows erects protective barriers around an Admin Mode program. Those barriers prevent it from interacting with regular User programs. But the pinball ecosystem is made up of a bunch of programs that were designed to interact with each other. So if you run program X in Admin Mode, and program Y needs to interact with it, then you *also* have to run program Y in Admin Mode. I think you can see where this idea that "you've got to just run everything in Admin Mode all the time" came from - it was a blunt instrument, but it was a way to get around these program interaction problems that Admin Mode created.

Okay, so if "Admin Mode everywhere all the time" is a simple way to solve thorny problems, why am I saying you *shouldn't* use it? The main reason is that, while it might solve some problems, it creates others. Microsoft doesn't want you to use Admin Mode routinely for everything, so you're always somewhat fighting with Windows if you do. It also reduces your system's security by defeating all of the protective mechanisms that Microsoft designed into Admin Mode in the first place.

The right solution - from a security perspective, and in terms of simplicity - is to stop using Admin Mode for *any* of your pinball software. If you run everything in regular User Mode, everything will be able to interact as it was designed to, with no hassles at the system level. Remember how I said that this whole Admin Mode fiasco is historical, because it was a requirement for certain components *in the old days*? Fortunately, it really is mostly relegated to the past now. Those old Admin Mode requirements were almost all due to software bugs, not actual engineering requirements, and all of the cases that I'm aware of have been fixed in modern versions. As of 2020, I don't think that any of the common pin sim components require Admin Mode, as long as you've updated to current versions.

If you do encounter any up-to-date pinball-related programs that say "Admin mode required", you should take a critical look at them and make sure the requirement is real, not just a misunderstanding. There's still a lot of confusion about this, so you can't always trust the FAQs and guides. My personal policy is that I simply won't run programs with unnecessary Admin Mode dependencies until the developer fixes them. I realize that not everyone can bring themselves to be so ruthless, when faced with a fun new feature that they really want. If you find a program that you can't live without, and there's just no way around its "requires Admin Mode" problem, I'd at least try to hold firm on one thing: don't let it "infect" the rest of your system with its Admin Mode requirement. One concrete thing you can do is to use PinballY as your front end. It has the ability to launch Admin Mode programs *without* running in Admin Mode itself. A major cause of the Admin Mode infection is that none of the other front ends can launch Admin Mode programs unless you also run them in Admin Mode, and of course if you do that, everything they launch will be in Admin Mode. And as I said, that might *seem* like it works for a while, but it's likely to eventually cause its own problems.

Customization log

Before you do anything else, I think it's a good idea to create "customization log" file. This is just text file for your own use - you can create it with Notepad and leave it empty for now. Put it someplace where you'll be able to find it easily in the future, such as right on the Windows desktop on your cab PC.

The point of this file is to jot down all of the special customizations you make to Visual Pinball and other software. VP in particular forces you to make some customizations in ways that you'll have to repeat each time you update to a new version. For example, some customizations require that you hand-edit VP's shared script files, and those changes will be lost on each update because VP will overwrite the scripts with its own updated copies. That's not a very friendly design on VP's part, I know, but it's just the way some things in VP work.

I'll mention this file again in other chapters when these sorts of changes come up, with a suggestion that you make a note in your customization log file. For now, just create the file so it'll be ready when you need it. In the future, whenever you make a change that warrants inclusion, add a note about it to the file. When it comes time to update VP or other software, you can refer back to this file to reinstate any customizations that got lost in the update process. The same goes if you ever have to rebuild your Windows system due to a system upgrade or disk failure.

Free pinball players

There are three main free pinball player programs for Windows:

- Visual Pinball 9
- Visual Pinball 10 (also known as VP X)
- Future Pinball

Visual Pinball is the essential program for a virtual cab. VP is an open-source project with an active developer community and frequent updates. Hundreds of tables are available, including re-creations of a pretty good percentage of all of the real pinball machines across the decades, plus many original tables. VP has excellent support for the whole gamut of special pin cab features: backglass monitors, DMDs, feedback devices, plunger inputs, accelerometer nudging.

I counted VP 9 and 10 as two separate programs because they're not compatible

with each other's tables, so you really have to install both. There's also a much older version 8, plus a couple of different, mutually incompatible versions of VP 9. Some people like to keep all of these installed because, again, individual tables are all tied to specific VP versions, so you need all of the VP versions if you want the ability to play all of the tables out there. (VP isn't very good at compatibility.) Fortunately, there's a combined installer that sets up the whole collection of VP versions with a single download and a single install process.

Future Pinball is another free player, but unlike VP, it's no longer being maintained or updated. Its original creator abandoned it a long time ago and never released the source code, so it's basically a dead end. Even so, you might want to install it to gain access to its tables, since there are a few re-creation tables (particularly from the 1970s or before) where there's an FP version but no VP version.

Visual Pinball 9 and 10

Visual Pinball 10, or "VP X", is the latest version, and VP 9 is the previous version. You'll want to install **both** versions because they're not compatible with each other's table files, and you'll want to be able to run both kinds of tables.

You can recognize VP 9 tables by the ".vpt" filename suffix. VP 10 tables use the ".vpx" suffix.

The easiest way to set up both versions is to use the VP Installer. VP is actually a collection of about five programs that work together, and in the old days, you had to go download them all individually and then go through a complex series of steps to configure them. The VP Installer bundles everything into a single download, and provides a Windows Setup program that configures it all automatically.

Here are the steps to install VP (both versions 9 and 10) with the VP Installer:

- Go to [vpforums](#).
- On the navigation bar near the top, click Getting Started. This will pop up a menu. Under "Install Visual Pinball", click "VP Installer".
- Even though this is called the "VP X Installer", it's actually the combined installer for VP 9 *and* 10.
- Click the Download button and follow the instructions to download the file. If you see several version options, pick the one with the highest number, since it should be the latest. You might need to create an account and log in before you can start the download.
- Unzip the downloaded file into a temporary folder on your hard disk.
- Double-click the Setup program.
- **Important:** when the program asks for a destination folder, use a folder in your hard disk's root folder, such as **C:\Visual Pinball**.

You can use a folder different name, but **don't** use anything within the Windows "Program Files" folder tree. Yes, that's the *normal* location for installing programs, but don't use it for VP. You'll create huge headaches for yourself if you do. The issue is that some VP components need to write files to their own install folders, and Windows has security restrictions against programs writing within the Program Files tree. The simple solution is to install VP somewhere else.

- If the program asks which DMD components to install, it's talking about the special "Dot Matrix Display" hardware devices that you can optionally install in your cab to re-create the plasma scoring display on 1990s pinballs. If you're

using a video monitor (such as a small TV or laptop display) for this, or you don't have a DMD panel at all, use the default option. If you're using a special external DMD device (PinDMD 2, PinDMD3, or Pin2dmd), select the corresponding option.

The VP Installer asks this question because each of the external hardware DMD devices require their own special software. The VP developers are working to combine all of this into a single unified system, which will eventually make it unnecessary to choose which one to use. If the installer doesn't ask this question, don't worry - it means you have a newer version with the unified software.

Future Pinball

Future Pinball isn't as essential as VP. It's an older system that hasn't been updated since 2010, and it's unlikely that it ever will be updated again, since its author abandoned the project without ever publishing the source code. I don't find its physics as convincing as VP's, and due to its age, FP's support for special cabinet features is limited.

Even so, many cab builders think FP is worth installing, since it's free and it has lots of tables available.

You can recognize tables written for FP by the ".fpt" filename suffix.

To install FP:

- Go to the Future Pinball site, futurepinball.com
- Click on the Download button near the top of the page
- Click on the Download link
- Run the downloaded .exe file, which will set up the program for you

Commercial pinball players

Some good commercial pinball games are also available. Here are the main commercial titles popular with cabinet builders:

- Pinball FX. A commercial pinball simulation available on Windows and other platforms. In 2018, this company acquired the Williams licenses that Farsight (see below) formerly held. They're gradually releasing table packs featuring re-creations of Williams/Bally/Midway titles. Pinball FX also offers a large collection of "fantasy" titles (original tables that never existed as real machines) from before they bought the Williams licenses, many based on popular media themes including the *Star Wars* movies and Marvel comics. Their older fantasy games had a decidedly unreal flavor, as they chose to fully embrace their video-game-ness by including elements that would have been impossible in a physical table. For some people that's a positive, since it makes the game action more diverse than in a real pinball machine, but it can be a negative if your tastes run more toward simulation and realism. Recognizing this, the FX developers say they've made changes to the physics engine in the new re-creations to make them play more realistically. This product has a Pin Cab mode available; to get it, you have to send a request to the publisher's tech support staff and provide proof that your cab is operated non-commercially.
- The Pinball Arcade by Farsight Studios. Detailed and accurate re-creations of real machines from the 1960s through the 2000s, available on Windows and

other platforms. TPA *formerly* boasted a large collection of Williams/Bally/Midway titles that included many of the best pinballs ever made. But Farsight's license to those titles was terminated in 2018 (to be taken over by the Pinball FX developers), so the editions you can buy now only include Gottlieb and Stern titles. Gottlieb dominated the EM era, so there are some great classics in there if you like the older machines, and Stern has been steadily producing newer machines since Williams withdrew from the market, many of which are popular and well-regarded.

The commercial games are playable on pin cabs, but they cater mostly to desktop users, and have limited support for pin cab features (DOF, multiple monitors, real DMDs, etc). Pin cab users aren't a big enough market to attract much commercial support, and of course the open-source developers who created all of the pin cab technologies are unable to modify closed-source commercial products.

Cabinet enhancements

Visual Pinball and the other pinball player programs are basically PC video games. To take full advantage of a cabinet, there are some additional pieces of software that you need.

Backglass display software

To display backglass artwork when playing Visual Pinball games, you need an add-on program called B2S Backglass Server. B2S is installed automatically along with VP if you used the VP Installer. If you set up VP manually, you'll have to install B2S separately.

Getting B2S working takes a few additional steps beyond just installing the software. We cover the details in Chapter 16, Backglass Software Setup.

Tactile feedback and lights

If you're installing any feedback devices in your cab - solenoids, shaker motors, flashing lights - then you need some additional software called DOF (DirectOutput Framework) to control the feedback devices.

DOF is an add-on program that lets Visual Pinball and other software access your output controller. DOF acts as the coordinator between the simulated game and the physical feedback devices, to synchronize feedback effects with the game action: firing your flipper solenoids when the flipper flips, activating the shaker motor when the castle is destroyed, etc.

DOF is a fairly big subject, so it gets its own chapter: Chapter 46, DOF Setup.

PinVol

PinVol is a utility I wrote to make it easier to control the audio volume during play. It lets you adjust the volume using cabinet buttons, and its special ability is that it helps equalize the volume level across different tables. It remembers your volume settings for each table individually, and automatically restores the table-specific settings whenever you switch tables. It has some additional special features for pin cabs, such as "night mode" (to reduce volume across all tables for late-night play) and individual level controls for multiple sound cards, all accessible from cabinet buttons.

You can find the download link and installation instructions on the [PinVol page](#).

Game selectors, or "front ends"

When your pin cab is finished, you'll probably want it to give the appearance of being a full-fledged arcade machine, not a plain old Windows PC. When you turn on the power, you won't want to see the Windows desktop at any point; you'll want something that looks more like a video game instead. It's also important to be able to operate all controls with the basic set of pin cab buttons - flipper buttons, Start, Exit.

This can all be accomplished with a program known on the forums as a "front end", so-called because it's the first thing you see when you walk up to the pin cab. A front end program serves as a replacement for the Windows desktop. It provides a video game-style user interface that lets you browse through your installed tables, launch tables, and switch between tables. A good front end will let you operate everything with the pin cab buttons so that you don't have to reach for the mouse or keyboard.

The most widely used front end currently is PinballX, which is free but closed-source. The original front end, HyperPin (also free-but-closed-source), is still around, but it's not very widely used any more; most people consider PinballX's user interface to be more modern and more pin-cab-friendly. There are also two newer options: PinUp Popper, another free/closed-source program; and my own PinballY, free and open-source.

PinballY

This is my own project, brand new in late 2018. I tried to make it easy and quick to set up so that you can try it out without a lot of hassle. It's designed specifically for pin cabs, and has built-in integration with most of the pin cab ecosystem, including Chapter 46, DOF, real DMD devices, joysticks (for button input), and multiple monitors. It's also highly customizable via a built-in Javascript scripting engine.

Downloads and more information are available at the [PinballY Project Page](#).

PinballY is similar to the other front ends in terms of user interface appearance and functionality. The main reason I wrote it was that I wanted an open-source option (all of the other front ends I know of are closed-source).

PinUp Popper

This is a newer program released in early 2018. It's free, but closed-source. See www.nailbuster.com/wikipinup/doku.php for download and install information.

PinballX

PinballX is currently the most popular front end for pin cabs. It has a minimalistic user interface that's well designed for pin cabs, letting you access all functionality with just four buttons (flippers, Start, and Exit), but also letting you use other buttons if you have them (e.g., MagnaSave).

You can download PinballX from its home site, pinballx.com. It's free to download, but it's closed-source, and installed versions "expire" after a period of time, requiring you to update. Follow the Download link from the main page to download the installer.

After running the installer program, you have to run the **Settings.exe** program in the PinballX folder. PinballX needs to know a bunch of things about your system before it will work properly. You should go through at least the Basic settings. Pay particular attention to the following:

- Display Settings page: Assign the monitors you're using for the playfield (which

PinballX calls the "main display"), backglass, and DMD (dot matrix display). Also set the rotations.

- Startup Settings: Set "Start with Windows" to Yes if you want the program to launch automatically when you boot the system.
- Keyboard Input Settings: set the key assignments to match the keys assigned to your cabinet buttons. If you're mapping the buttons to joystick buttons, you can assign those on the next page, Joystick Input Settings.
- Future Pinball, Visual Pinball: Set the directory paths for these programs. The "Working Path" field should be set to the folder containing each program.

Adding tables to the PinballX menu

PinballX doesn't go out and find your tables by itself. You have to enter each table into PinballX's menu list yourself. You do this using the Game List Manager program in the PinballX program directory. Before running this, make sure you configured the directory locations with the PinballX Settings program as described above.

The PBX installer will pre-populate the menu list with a few games for demo purposes, so the first thing you'll probably want to do is delete these. Simply click the Delete button next to each game in the list. Note that there are multiple game lists (Visual Pinball, Future Pinball, MAME), so you'll have to select each list with the drop list at the top of the window and delete its games.

Each pinball game you set up has a bunch of associated "media" items: a "wheel" image, which provides the title graphic shown in the menu when you navigate to the table; a playfield image; a backglass image; a DMD image; the advertising flyer for the game; an instructions card; video versions of the table and backglass images; and audio to play when you launch the game. You can set up each of these items individually, but that's extremely tedious, especially if you have lots of games to add.

Fortunately, there's an easier way.

The quick way to set up a game is to use the "Import Media Pack" button at the top. This lets you add a game, along with all of its related media items, in one operation. You'll still need to select the game's playable file (the .vpx file for VP 10, for example), but everything else will be set up automatically.

To set up a game using the "import" button, start by downloading the game's media pack. You can find media packs on vpforums.com. Select "Frontend Media & Backglass" on the navigation bar, then click "Complete Media Packs" under the Media Packs section. This will take you to a gigantic list of "HP Media Pack" files. The "HP" is for HyperPin, but PinballX knows how to read these same files. Navigate through the list to find the game you're looking for.

Each of these "HP Media Pack" files is an ordinary ZIP file. Don't unpack them. Simply download them to the Tables directory for the appropriate pinball player version. For example, if you're setting up a Visual Pinball 10 game, download the corresponding table pack to the Visual Pinball 10\Tables folder.

Now go to the PinballX Game Manager. Select the list for the appropriate pinball player at the top (e.g., select "Visual Pinball"). **Don't click Add Game** at any point. Instead, click Import Media Pack. Select the ZIP file you downloaded. This will automatically create a new entry for the game and populate it with the media items in the ZIP file. Now click on the Select button next to the Game field for the newly added item. Choose the playable game file from the list. Note that this will only show you a list of game files you've already installed in the Tables folder, so you'll have to actually download the game into the Tables before you can complete this step.

After you exit out of the Game Manager program and restart PinballX, you should now see the newly added game show up in the menu.

As you add tables to your system, you'll need to repeat this process for each one.

HyperPin

HyperPin was the original front end for pin cabs. It's an offshoot of the similarly named HyperSpin, which is a popular front end for home-brew video game cabinets. Since HyperPin came from the video game world, it was designed around an assumption that you have a big bunch of buttons. Pin cab builders tend to prefer a more minimalistic approach, with only a small set of buttons closer to what's found on most real pinball machines. This has always made HyperPin a little ill-fitting on a pin cab, since its UI depends on having a fairly large number of buttons that can be mapped to individual functions. A lot of early pin cab builders designed their cabs specifically for HyperPin by installing four or five extra buttons on the front panel dedicated to front-end functions. But most of us don't like the extra buttons on aesthetic grounds, because they take away from the real pinball look. That's a big part of why so many pin cab builders migrated to PinballX when it became available.

The home site for HyperPin is hyperspin-fe.com. Click the Download button in the main navigation bar, then look for "HyperPin" in the Category list.

Where to find tables

Visual Pinball tables: The biggest collection I've seen of VP cabinet-mode tables is [vpforums](http://vpforums.com). Click "Visual Pinball Tables" in the navigation bar at the top. The popup menu has several sections; the ones you'll want to look in for pin cab use are "VP9 Cabinet Tables" and "VPX Tables" section. VP 9 requires tables to be designed specially for cabinet use, which is why it has a special section. VP 10 unifies cabinet and desktop modes, so it doesn't have a separate cabinet section - any VPX table should work in cabinet mode.

[vpuniverse](http://vpuniverse.com) also hosts VP tables, although their collection isn't as extensive. Click the Downloads link in the navigation bar to find tables.

Future Pinball tables: As with VP 10, all Future Pinball table files are playable in cabinet mode. You just have to adjust the camera settings for each table to get it lined up properly for cabinet play. [vpforums](http://vpforums.com) has a large collection of FP tables: click "Downloads" in the navigation bar, then look in the "Future Pinball Tables" section.

Backglasses: Some tables include the B2S backglass files with the Visual Pinball table files, but most don't, so you'll usually have to download backglass files separately. [vpforums](http://vpforums.com) has a large collection of these: click "Frontend Media & Backglasses" on the navigation bar, then select "dB2S Animated Backglasses" under the Backglasses section.

PinballX & HyperPin media: [vpforums](http://vpforums.com) has a large collection of media packs for the front-end menu program. Click "Frontend Media & Backglasses" on the navigation bar, then select "Complete Media Packs" from the "Media Packs" section. "HP Media Pack" files work in both HyperPin and PinballX.

16. Backglass Software Setup

If your pin cab has a separate backglass monitor, you'll want it to display the appropriate backglass artwork for the current game. And you'll want this to be more than just a still image, since the backglass is an active part of many games, showing information on score, bonus features, etc.

Fortunately, this is well supported in Visual Pinball. VP can display live, animated backglass artwork that synchronizes with the game action. VP requires an add-on program called B2S Backglass Server to do this. B2S works alongside Visual Pinball to display the animated backglass artwork, simulating the same backglass lighting effects, score displays, and animated elements that you'd see on a real machine. B2S is specifically designed for a cabinet setup where you have a separate monitor for the backglass.

To get all of this working, you have to install the B2S software itself, and then there are some extra setup steps required for each table. This chapter explains how to set up the B2S software and configure tables to use it.

B2S Installation

The first step to getting backglasses working is to install the B2S software itself.

If you installed Visual Pinball using the VP Installer program, B2S should have been installed automatically as part of the setup process, so you're already set.

If you didn't use the VP installer (that is, you installed VP manually from ZIP files or something like that), you'll have to install B2S separately. To find the download:

- Go to vpforums.org
- Click **Getting Started** the top navigation bar
- Select **Essential Files > Frontends and Addons**
- Find "B2S Backglass Server" in the file listing

There might be several versions of B2S Backglass Server in the file list. I'd recommend picking the one with the highest version number to make sure you have the latest update. Click the link, which will take you to the download page. That should contain links for downloading the file and for installation instructions. Note that you'll have to create a vpforums account to download a file, if you don't already have one.

Click the "instructions" link on the download page and follow the steps. Here's the basic procedure:

- Unzip all of the files from the downloaded B2S ZIP file into your Visual Pinball\Tables folder.
- Right-click the file B2SBackglassServer.dll. Select Properties from the menu. Check for a message under the "General" tab saying something like "This file was downloaded from the Internet and has been blocked." If you find this, there should be an "Unblock" button. Click it. If you don't find any such message, no action is required.
- Right-click the application file B2SBackglassServerRegisterApp.exe. Select **Run as administrator** from the menu.
- Check the README.TXT file from the downloaded ZIP file for any additional instructions or notes for the version you downloaded.

Download backglass files

Okay, you've installed B2S, loaded up a game in VP... and didn't see any backglass artwork. That's because installing B2S isn't the last step. You also have to do a little extra work to set up each table. This is work that you'll have to repeat for each new table you download, but fortunately it's only a one-time job for each table.

Most VP table files are distributed without any backglass artwork. Some authors bundle the two together, but in most cases you have to download the backglass file separately.

Fortunately, it's fairly easy to find backglass files. vpforums.org has a large collection:

- Go to vpforums.org
- Click **Frontend Media & Backglass** on the top navigation bar
- Click **Backlasses > dB2s Animated Backlasses**

Files are listed by table title, so just find the title of the table you're looking for and click through the links to download the file.

Are B2S files and .vpt/.vpx files paired?

No. You **don't** have to find the exact matching B2S version for your table. Any B2S for a given title should work with any .vpt/.vpx game for the same title. E.g., you don't need a specially paired B2S for "Funhouse_NightMod_ToyMod_991_v26.vpt"; any B2S for Funhouse should work.

This is the whole reason that the table files and backglass files are usually distributed separately. The B2S files and VP table files are more or less independent in terms of their operation and design. A particular .vpt or .vpx file should work with just about any .directb2s file for the same table. There's no need to find a matching set for a particular version of either.

2-screen and 3-screen versions

Some of B2S backlasses have "2-screen" and "3-screen" variants. The difference is how the speaker/DMD panel graphics are handled:

- A 2-screen backglass includes graphics for the speaker panel as part of the backglass window. This is ideal if you have a single large monitor in your backbox and no separate DMD video monitor or real DMD device.
- A 3-screen backglass separates the graphics for the backglass area and speaker panel area into different windows, so that you can position the two areas separately on your physical monitor layout. This is ideal if you have a 1990s-style backbox, with a 16:9 monitor at the top, and a separate speaker panel with its own DMD video monitor or real DMD device.

In cases where both types are available, choose the one that matches your physical cab setup. But it's also okay if only one type is available and it's the "wrong" one for your cab. You can always use either format. The worst that happens with the "wrong" format is that the geometry will be a bit distorted because the proportions were designed for a different monitor layout.

Installing the .directb2s file

After downloading a backglass file, perform these steps:

- Unpack it from the ZIP/RAR container if necessary
- Put it in your Visual Pinball **Tables** folder (the same folder where you keep your .vpt/.vpx files)
- Rename the file so that it matches the name of the .vpt/.vpx file you want to use it with, but keeping the .directb2s suffix (e.g., if the table file is called Funhouse_NightMod_v2.vpx, rename the B2S file as Funhouse_NightMod_v2.directb2s)

It's essential to put the .directb2s file in the same location as the table file and to rename it to match the .vpt/.vpx file name. That's how B2S finds the file. If the name and location aren't matched like this, B2S can't usually locate the file and won't show any backglass artwork.

(B2S does actually have some "fuzzy matching" that tries to find a matching file even if the name doesn't exactly match, but in my experience, that does more harm than good. B2S's fuzzy matching usually just picks something random and wrong, and the fact that it's doing it at all makes it that much harder to figure out why the wrong file is getting picked. The only way to make B2S pick the right file reliably is to give it the exact same name as the table file.)

Enable B2S on each table

There's one more step before the backglass artwork will appear during a game: you have to enable B2S mode for each individual game. This is a one-time step, but you have to do it separately for each table.

Most VP 10 tables will automatically use B2S if present, so this step isn't usually required for VP 10. However, if you try a VP 10 table and it doesn't work, it might be an unusual case that requires the VP 9 procedure below.

Some later VP 9 tables, from 2016 and later, also will automatically work with B2S.

Given that most VP 10 tables and some VP 9 tables will work "out of the box" without any modification, the first step is to simply fire up the game in VP and see if the backglass appears. If so, you're all set with that game. If not, try this procedure:

- Open the table in the VP editor (see Chapter 18, Customizing VP Tables)
- Open the table script (VP 9: **Edit > Script** menu command; VP 10: **View > Script** command)
- Look for a line like this:

```
Const cController = 0 ' 1=VPinMAME,
                      ' 2=UVP backglass server,
                      ' 3=B2S backglass server
```

- Note that the variable name cController might be slightly different, so just look for something that roughly matches that format.
- If you find such a line, change the "0" to the number listed for B2s (usually 3). Save and run the table. If it successfully displays the backglass, you're set.
- If you can't find the code above, look for something like this:

```
Set Controller = CreateObject("VPinMAME.Controller")
```

- If you find that, replace it with this:

```
Set Controller = CreateObject("B2S.Server")
```

- Save the game and try again.

If none of the above helped, and the table is a re-creation of an older EM (electro-mechanical) game from the 1970s or earlier, the game will need more extensive modification to make it work with B2S. The work needed is beyond the scope of this chapter. Your best bet might be to contact the author of the table and request a B2S-capable update, or see if someone else on the forums wants to take it on.

17. Optimizing Performance

Virtual pinball is in essence a video game, and video games place special demands on PC performance. The first prerequisite for adequate performance is adequate hardware, particularly the CPU and graphics card. You can find advice on selecting suitable components in Chapter 10, Designing the PC.

Even with fast hardware, though, you'll need to do some tuning to get the best video game performance out of your system. This chapter offers some advice on things to try.

Here's my quick list of the most fruitful optimizations with VP:

1. Minimize background tasks and other running programs (see General Windows optimizations)
2. Use a CPU affinity tool to give VP exclusive access to a group of CPU cores (see Controlling CPU affinities)

How to approach system optimization

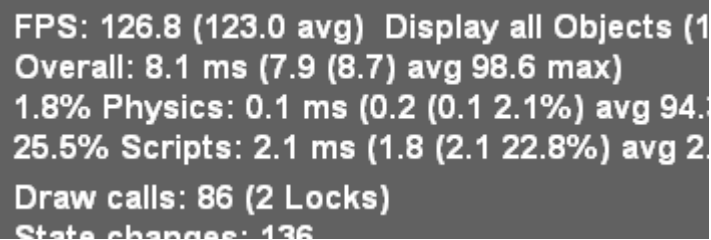
The whole point of optimizing is to make the game play better, in your subjective view. It's worth making a point to keep that in mind, because it's easy to get bogged down in the numbers, trying to make specific benchmark figures or performance metrics as perfect as possible. It's best not to get too obsessed with any one benchmark, because there's no benchmark that captures everything, and there's always a point of diminishing returns when you start focusing on making one number as high as you can get it.

That said, objective metrics are extremely helpful when making adjustments. For many of the adjustments you can make, there's no one-size-fits-all ideal setting, since the effects vary by system, so you'll have to experiment in many cases to find the right setting. You'll also have to experiment to see which types of adjustments make any sort of difference. Objective numbers can be really helpful to see if adjustments are having any effect and if they're moving the needle in the right direction. It's easy to trick yourself into seeing differences that aren't there if you don't have some kind of hard data to look at.

Measuring VP performance: frames per second

The most common performance metric in video games is the "frame rate", in Frames Per Second or FPS. This is the number of times per second that the game software can fully render the scene and update the video graphics.

VP can display the current frame rate and some other statistics on-screen while you play. Activate this display by pressing F11 while a game is running.

A screenshot of a performance statistics overlay from the game VP. The text is white on a dark, semi-transparent background. It displays various performance metrics including FPS, overall time, physics, scripts, draw calls, and state changes.

FPS: 126.8 (123.0 avg) Display all Objects (1
Overall: 8.1 ms (7.9 (8.7) avg 98.6 max)
1.8% Physics: 0.1 ms (0.2 (0.1 2.1%) avg 94.
25.5% Scripts: 2.1 ms (1.8 (2.1 22.8%) avg 2.
Draw calls: 86 (2 Locks)
State changes: 136

At first glance, it might seem strange that this isn't identical to your video card's

refresh rate, usually 60 Hz (updates per second). The reason for the difference is that most video games (VP included) do their graphics rendering on their own schedule, according to how quickly they can do the computing work to produce each frame.

In order to produce smooth animation, VP's frame rate has to be at least as fast as the hardware refresh rate. If VP can't produce a new frame in time for the next hardware refresh, the TV will keep displaying the same frame as on the last refresh. This will make the picture momentarily freeze on the TV, which makes the motion appear jerky.

In practice, it's not good enough for the VP frame rate to merely be higher than the TV's refresh rate. It has to be *much* higher, by a factor of two or more. This is because the FPS rates that VP shows you is an average over many frames. The actual time it takes to produce any individual frame can vary quite a lot from one frame to the next. Some frames might take two or three times as long as the average frame to produce. If your average FPS rate is only marginally above the TV's refresh rate, all of those slower-than-average frames will take longer than one refresh cycle to generate, so you'll see a lot of repeated frames and jerky motion.

What makes a frame take longer than average? It's a matter of the complexity of the physics and graphics models that go into making up the frame. You'll typically see the frame rate slow down during multiball sequences, for example, since the physics engine has to compute the motion of the additional balls.

How to check your graphics card refresh rate

- Open the Display control panel in Windows
- Go to the Adjust Resolution section
- Click Advanced settings
- In the Adapter tab, click List All Modes

This will show you the list of screen resolutions and refresh rates that your GPU supports.

Most graphics cards and TVs use a standard 60 Hz video refresh rate. Many LCD TVs have nominal refresh rates of 120 or 240 Hz, but this isn't the video signal rate; it's just the LCD panel update rate, which uses interpolation to synthesize fake frames between the actual frames. A small number of TVs and computer monitors can accept true video signal rates higher than 60 Hz, but to take advantage of that you also need a video card that can generate such signals.

It's always best to use a graphics mode that exactly matches the native resolution of your TV: 1280x720 for a 720p TV, 1920x1080 for a 1080p TV, or 3840x2160 for a 4K TV.

How to reduce stutter

The jerky motion that happens when frames are repeated due to slow rendering is called "stutter". It's more or less impossible to make VP absolutely stutter-proof, since there are occasional oddball situations in any game where the physics computations get extremely complex and overwhelm even the fastest hardware. But it's at least possible to reduce stutter to the point where you'll hardly ever see it.

There are really two separate sources of stutter, which require separate solutions.

The first source of stutter is the raw computational time that VP spends updating the

physics and rendering the graphics. The physics updates are done on the main CPU, and the rendering is mostly done on the graphics card (the GPU). So the obvious route to faster updates is faster hardware. For most people, that's something to consider in the planning stages, but it's not practical in terms of budget to update your whole PC every time a table is too slow. Barring hardware updates, the main way to increase VP's raw rendering rate is by adjusting VP's graphics options. We'll cover that below in VP video settings.

The second source of stutter is a little more subtle. Windows multi-tasks - runs many programs at once - by letting programs take turns using the CPU. Windows lets each program run for a fraction of a second, then interrupts it and lets another app take its turn. With most applications, you never notice this turn-taking, because it happens so quickly that it creates the illusion that every program is running at the same time. The illusion can start to break down with video games, though. The problem is that the turn-taking interruptions can happen at inopportune times that make VP miss its window for updating the graphics in time for a video refresh cycle, causing repeated frames and thus stutter.

One obvious way to reduce these interruptions is to minimize the number of other running programs, as outlined in General Windows optimizations below. That only goes so far, though, as you can't shut off everything else. To deal with the programs that have to keep running, there's a powerful technique known as CPU "affinity", which lets you partition your hardware and give special VP special access to parts of it. We'll talk about that later in Controlling CPU affinities.

General Windows optimizations

You can find lots of advice on the Web about general Windows tuning and gaming PC tuning. There's so much advice of that sort available that I won't try to reiterate it all in detail here, other than to mention the main points:

- Delete unused programs, especially any bloatware pre-installed by the PC vendor
- Disable unnecessary startup programs
- Remove or disable unnecessary background programs and Windows "services"
- Disable programs running in the "system tray" (the little icon area in the Windows task bar near the clock)
- If you're using Windows 10, turn off all of the cloud features in the system "Privacy" settings, to minimize background network access.
- Remove third-party antivirus/antimalware software

The last one (removing third-party antimalware programs) might make you uncomfortable. It's up to you, obviously, but I think it's worthwhile for a machine that you use as a dedicated pin cab PC and not as your main PC. There are two reasons I think you can go without. First, Windows 7 and later have pretty good protection built-in, in the form of Windows Defender. That's positioned for marketing purposes as "basic" protection only, to give you the impression you need something stronger, but it actually scores fairly high in most independent testing. Second, you can greatly reduce your exposure to malware by using the pin cab PC exclusively as a pin cab PC: don't store sensitive personal files on it, don't use it for email, don't use it for random Web browsing (limit Web use to trusted sites), and don't download random files (only download from trusted sites). Email and random Web browsing are the primary vectors for malware, so you can minimize exposure to avoiding those vectors as much as possible.

Controlling CPU affinities

The most powerful tool I've found for reducing stutter is CPU affinity. This is a mechanism inside Windows for assigning each running program to a preferred group of CPU cores.

A "core" is a CPU within your CPU. The processor chips used in modern PCs, such as Intel i5 or i7 chips, are actually made up of multiple CPUs packed onto one piece of silicon. For example, an i5-8250 chip contains four complete CPUs. The term "core" is used to distinguish these CPU sub-units from the chip as a whole, which is also commonly called a CPU.

Windows has built-in support for multi-core chips. It automatically spreads work across the cores to optimize overall system throughput, and for most purposes you don't even have to think about it. As usual, though, video gaming doesn't exactly fit the typical program profile that the Windows default settings are designed for. The core affinity feature in Windows lets you override the defaults to optimize performance for special cases like games.

CPU affinity goals

The basic idea is to partition your CPU's cores into two groups: VP, and everything else. When you're running a game in VP, the game itself is the only performance-critical task in the whole system; everything else can take a back seat and wait its turn. So we're going to give the lion's share of your PC's computing power to the game, and give all other running programs the leftovers. For CPU affinity settings, the smallest unit we can work with when dividing things up is one CPU core, so if your CPU has N cores, we're going to allocate N-1 of the cores to the game, and give the one remaining core to everything else.

The point of this partitioning is to give the pinball software the most exclusive access we can to a set of CPU cores. This reduces the chances that another program running in the system will be able to interrupt VP or its components in the middle of some time-critical tasks. (And virtually everything VP does is time-critical, since it's a real-time physics simulation.)

In practice, I find that this makes a night-and-day difference in stutter, reducing it from noticeable to practically never on my pin cab.

CPU affinity tools

PinAffinity: This is a simple CPU affinity setter I wrote specifically for pin cabs. It's designed to be extremely simple to set up and completely automatic once configured, and it's free and open-source. You can find it here: [PinAffinity](#).

Instructions for basic pin cab setup are included in the download, but here's a quick overview:

- Download the "bit" version that matches your copy of Windows (32-bit or 64-bit)
- Unzip the files into a folder on your hard disk
- Run PinAffinity.exe
- Use the "Add Program" menu to add the .EXE file for each pinball player program on your system to the designed Pinball program list
- Minimize the PinAffinity window and leave it running in the background while you play. It automatically sets CPU affinities for new processes as they're created.

- If you wish, you can create a shortcut to PinAffinity.exe in your Start Menu "Startup" folder so that the program automatically launches each time you boot

Other tools: On my own cab, I used to use a freeware program called PriFinitty. Unfortunately, it's no longer available; the developer abandoned the project a long time ago and never released the source code.

Another option is Process Hacker 3, available here: wj32.org/processhacker/. Process Hacker is a full Task Manager replacement, so it's not specifically designed for the pin cab use case, but it has the basic function we need (the ability to set CPU affinities persistently on a per-program basis). Note that you'll need a "nightly build" version of Process Hacker 3. The current public release version, Process Hacker 2, can control CPU affinities for live processes but can't save them or apply them automatically to new processes.

Recommended CPU affinity configuration

I recommend the following basic configuration:

- Assign three cores to VP (and any other pinball software you use)
- Assign the remaining cores to everything else

PinAffinity uses those settings by default.

Why three cores for VP? You might have read that Visual Pinball is single-threaded, so it might not seem like it would benefit from more than one core. It's true that VP's core physics and graphics run on a single thread, but if you look at a VP process with a tool like Process Explorer, you'll see that it has about 20 threads running. Where are they all coming from if VP is single-threaded? Mostly from the external subsystems that VP uses. VPinMAME runs on its own thread; DirectInput and DirectSound create multiple threads to service I/O events; DOF creates a thread for each output controller it accesses; and most video card graphics drivers create several additional threads.

The main VP physics/rendering thread and the VPinMAME thread each consume significant CPU time; the rest of the threads do little actual computing work, as they only exist to service I/O events and timed events as they occur. The VP and VPM threads probably don't come close to saturating the CPU on your machine; you'll probably see only 10% to 20% CPU usage on these threads. But even so, they both benefit from having a free core available because they both need "real time" responsiveness to keep up with external events. In the case of VPinMAME, it has to respond to game events immediately when they happen, and has to maintain precise time sync with the audio playback to prevent audible glitches in the soundtrack. In the case of VP, the physics/rendering thread has to keep in precise sync with the video refresh cycle; any lag in rendering is visible as stutter. It also has to respond quickly to input events to avoid perceptible latency (e.g., so that the flippers don't feel sluggish when you press the buttons).

That's why three cores seems to be the sweet spot for VP performance. We have two threads that run more or less continuously (the main VP physics/rendering thread, and the VPinMAME thread), and a bunch of other threads that need to respond quickly to events but do little work. If you give this collection of threads three cores to work with, Windows will be able to balance the load so that everything has near-real-time CPU access.

One video card or two?

One of the frequently asked questions on the forums is whether it's better to use a single video card that can support multiple monitors, or a separate video card for each monitor. (Most pin cabs have either two or three monitors: the main playfield TV, the backglass TV, and possibly a third monitor for the score display or "DMD" - the dot matrix display.)

This question actually contains two components, so let's unpack it. The first part is: can my video card handle multiple monitors? The second part is: would I get better performance by adding an extra video card for the second and third monitors?

The answer to the first part is basically always yes. All modern gaming cards have support for multiple monitors and provide built-in ports for connecting two to four monitors. And Windows has excellent, fully automatic support for multiple monitors built in. When you're setting up your system, there shouldn't be anything special you have to do with either your video card or Windows to configure multiple monitors; you shouldn't have to do anything more than just plugging them all in.

What about performance, though? Intuitively, it seems like more video hardware should translate to faster performance, for the same reasons that CPUs with more cores run faster. It seems like an extra card would take some load off the main graphics card. In practice, though, adding a second card makes most systems run slower. I can only speculate about the reasons for this, but I suspect that it has to do with contention for the data bus that connects the CPU and GPU. A second video card creates bus contention that doesn't exist if there's only one video card. Whatever the reason, most system in practice run faster with a single video card handling all monitors.

"One card is faster" isn't an absolute rule, though. I have heard from people who found that adding a second card actually did improve performance on their systems. But this seems relatively uncommon; for most systems, it seems that you'll get better performance by buying one fast video card than by trying to split the load across two or more cards.

Input lag

A common performance problem in video games is input lag: a noticeable delay between pressing a button and seeing the result on-screen. On a pin cab, this is mostly noticeable with the flippers. Input lag makes it feel like the flippers are slow to respond when you push the buttons.

Latency can come from many sources, but in most cases, the culprit turns out to be the TV. That's the place to focus your efforts to fix the problem, in part because it's almost always the biggest contributor to lag by far, and in part because there aren't really any adjustments to be made anywhere else in the system. Everything else is probably already running as fast as it can.

You can find more about input lag and how to minimize it in Chapter 7, Selecting a Playfield TV. We won't repeat all of the detail here, but the main point is that you should be sure your TV is set up with as little image processing as possible, especially enhancement modes related to "motion smoothing". Look for a Game Mode setting in your TV's setup menus; that usually selects the right combination of settings to minimize lag, since this is a concern for video games in general.

Audio lag

Audio lag is a noticeable delay between visual events appearing on-screen and the playback of the corresponding audio effect. This fortunately isn't a common problem.

If you experience it, though, here are a few things to try.

Check your sound card's settings. (Open the Windows "Sound" control panel, select your card, and click Properties.) Make sure that everything is turned off under the Enhancements tab.

Some sound cards have extra properties tabs that control special hardware features of the card. Check any extra you see and make sure that any processing modes, effects, or enhancements are disabled.

If you're using an add-in sound card (rather than motherboard audio), and it came with its own separate settings program, go through that and make the same kinds of checks: look for any effects or processing modes that might be slowing things down, and disable any you find.

If you're using your TV's built-in speakers through the HDMI video connection, check the setup menus to see if there's a setting specifically for audio delay or audio sync. Your TV might be intentionally delaying the audio signal so that it syncs up with delays in the video signal processing, and it might let you adjust the delay time. If so, make it as short as possible.

VP video settings

Visual Pinball has a number of options related to graphics rendering that can affect performance. To access these options, start VP without loading a game, in editor mode. On the menu, select Preferences > Video Options.

The effects on performance of the different settings vary by system, and many of them trade off between quality and speed, so you'll have to experiment to find the ideal settings for your system. Here's an overview of the main settings that affect performance:

- Anti-aliasing: this controls extra graphics processing to make edges look smoother. Disabling it does the least processing, so it will be the fastest, at the expense of rougher looking edges. The performance impact of the different anti-aliasing modes depends on your video card.
- Ball reflections, ambient occlusion: these control extra rendering to create more realistic graphics. Turning them off results in faster rendering times.
- FPS limiter: setting this to 0 (the default) allows VP to render video frames as fast as it can. Setting it to 1 makes VP synchronize its rendering cycle with the actual video refresh rate. Some people find that synchronized rendering produces smoother graphics, so you might want to try it to see if it makes any noticeable difference for you. If not, I'd leave it at the default 0 setting.
- Exclusive full-screen mode: this makes VP take over the monitor completely when running in full-screen mode, rather than sharing it with other programs. This can improve graphics performance on some systems, but it can cause weird glitches with Windows multitasking, so I'd avoid it unless it makes a big difference for you.

18. Customizing VP Tables

One of the great things about the Windows virtual pinball environment is that so many tables are available in a format that lets you modify and customize them in any way you please. The free pinball player programs (Visual Pinball, Future Pinball) are also full pinball construction programs. You can open any table for Visual Pinball or Future Pinball in its editor and make your own modifications.

(The same can't be said of the commercial pinball games, such as Farsight's Pinball Arcade or Pinball FX/2 and FX/3. With those, you're stuck with what they sell you, which is never a perfect fit for cabinet play. That's one big reason that VP is so popular with pin cab builders.)

This chapter provides an overview of how to customize tables in Visual Pinball. We go into detail on a few of the key customizations often needed to adapt a table to cabinet play. Many of the tables available for VP were designed with regular desktop in mind, and weren't tested on a cab by their original authors, so they need some tweaking to look and play their best on a cab.

Opening a table in the VP editor

Visual Pinball is, at its core, a pinball construction program that also happens to let you play the tables. So editing a table is really the default thing that VP does.

In VP 8 and 9, the first thing you see when you start the program is a blank table editor window. Use the **File > Open** menu command to open a table in the editor.

In VP 10, they tried to make VP a little friendlier for the "average user", who just wants to play existing tables rather than creating their own. So VP 10 starts by bringing up a "Select Table File" dialog when you first start the program, and then immediately starts a game session with the table you select. If you want to edit a table instead, you have to bypass this initial dialog. Just click Cancel in the dialog, then use the **File > Open** menu to open a table in the editor, just like in VP 8/9.

Adjusting the viewing angle

When VP runs a game, the image you see on the display is a rendering of a 3D model of the table. To construct this image, VP uses an imaginary camera that views the model from a selected position in space. You can adjust this camera position to create different views of the table.

I find that most of the VP tables I download need some adjustment in their viewing angle to look their best on a pin cab display. And most pin cab builders feel the same way, because the ideal viewing angle is subjective. No one viewing angle will satisfy everybody.

VP 9: Adjusting the viewing angle in VP 9 is a bit tedious because you have to do it by typing numbers into a property sheet in the editor, and then run the game to test the results. I always have to iterate this process five or ten times before I find a satisfactory solution.

- Open the table in the VP editor
- On the left tool palette, click the **Backdrop** button
- Make sure the properties panel on the right is showing; if it's not, click the **Options** button on the left tool palette
- The properties panel should be labeled **Backdrop** at the top; if it's not, click in

a background area of the editor window to select the backdrop

In the Backdrop properties window, the **Colors & Formatting** section contains all of the viewing angle options. The exact settings vary a lot from one table to the next, so there's no one-size-fits-all setting list I can give you. You'll just have to experiment with the different settings to see the effect they have. Change a setting and run the table to see the result. Change one thing at a time so that you can see each setting's individual effect.

Here's an overview of the individual settings and what they do:

- **Inclination:** The camera tilt, in degrees. 0 points the camera straight down at the table. Positive values tilt the camera upwards. For cabinet use, this should usually be close to 0.
- **Field of view:** The camera's viewing angle. This is analogous to a zoom lens on an optical camera. Zero produces an extremely flat view, like a telephoto image from a long way away; a high number (120-150) creates an exaggerated fisheye lens view. A value around 20 is usually good for cabinet use.
- **Layback:** The camera's distance from the front of the table. Higher values create a more tilted perspective. 0 creates a view from right over the center of the table. You want a value that places the camera a little ways out in front of the table, just like the normal viewing position for a player. A value of about 2/3 of the Y offset below is usually good.
- **XY Rotation:** For cabinet play, set this to 270.
- **X Scale:** This adjusts the table's size relative to the width of the monitor, which is confusingly the height of the table, when rotated 270 degrees for cabinet play. Adjust this to fit the table in the monitor across the monitor's width. This varies a lot by table, since it's a function of the playfield dimensions as well as the camera angle settings above. A value of around 1.4 works for many tables, but you'll have to fine-tune it for each table.
- **Y Scale:** This adjusts the table's size relative to the height of the monitor, which is the width of the table, when rotated 270 degrees for cabinet play. Adjust to fit. A value of around 2.0 works for many tables, but you'll have to fine-tune it for each table.
- **X Offset:** This adjusts the side-to-side position of the table, which is confusingly the vertical position on the monitor, when rotated 270 degrees for cabinet play. Adjust this so that the table is positioned properly. A value of around -450 works for most tables, but you'll have to fine-tune it for each table.
- **Y Offset:** This adjusts the top-to-bottom position of the table, which is confusingly the horizontal position on the monitor, when rotated 270 degrees for cabinet play. Adjust this so that the table is positioned properly. A value of around 50 works for most tables, but you'll have to fine-tune it.

VP 10: You can use exactly the same procedure as above with VP 10, but VP 10 also has an interactive "camera mode" that's a little easier to use. Camera mode lets you see the effect of each change immediately on the rendered table, without having to switch back and forth between editor mode and play mode repeatedly.

To activate camera mode, use the menu command **Table > Camera/Light Edit Mode**, or press F6. Follow the on-screen instructions to cycle through the settings and make adjustments. The settings listed above for VP 9 all have the same meanings here.

The camera mode controls are a little cumbersome, and it's hard to set exact values with them. You can always go back and fine-tune the values with the properties editor (using the VP 9 procedure above) to make any final adjustments.

Fake 3D table elements

One thing to note is that a lot of tables have some "fake 3D" table elements that don't respond well to viewing angle adjustments.

For example, consider Rudy's head in *Funhouse*. On the real machine, of course, Rudy is a rather large 3D chunk of plastic. But some VP versions of *Funhouse* don't use a 3D model object for Rudy; they just use a photo of Rudy pasted onto a flat surface in the VP model. That's what I mean by a "fake 3D" element: it's meant to look like it's 3D, but it's actually just a flat photo in the software.

The problem with these fake 3D objects is that the viewing angle captured in the photo will stay the same no matter how much you change the viewing angle of the table. The photo is, after all, just a photo. If you change the overall table viewing angle too far, it will become extremely obvious that the flat photo is now from the wrong perspective.

There are a few ways you can deal with this when you run into it:

- You can live with the distortion. The distortion will become more obvious the further you change the table viewing angle, so if you only need to adjust the angle a little bit, the distortion might remain tolerable.
- You can take or find a new photo from the new angle and replace the one in the table. This is tough unless you have access to the real machine, but you might get lucky and find a suitable image on the Web. You can find a lot of images on the Web for the more popular titles, after all.
- You can substitute a real 3D model (known in VP parlance as a "primitive") for the fake, flat photo. Ask on the forums to see if someone has already created one; there are 3D models of lots of pinball elements floating around (even unique ones like Rudy's head). Browse through some generic 3D model sites looking for something similar that you can adapt via Blender or SketchUp. If it's not too complex, create one yourself with one of those programs.

Once you have a 3D object, you have to save it in the Wavefront ".obj" format. This is a common format that most 3D editors can save to. Next, create a "primitive" object in VP and import the .obj file. You'll also need a "texture" (an image that's projected onto the 3D surface to provide its coloration). The details are beyond the scope of this guide, but you should be able to get help in the forums if you're not familiar with VP primitives.

Viewing and editing the table script

Many customizations in VP are made through the table's "script". Every table has a script, which is basically a little computer program that carries out certain operations when you're playing a game with the table. It's called a "script" by way of analogy to the script for a movie or play. A movie script is a series of things the actors are supposed to say and do during the movie; a VP table script is a series of things the computer is supposed to do while the the is running.

To view a table's script:

- In VP 8/9, use the **Edit > Script** menu command
- In VP 10, use the **View > Script** menu command

That brings up a text editor window showing the script. You can simply type into this window to edit the code.

Table scripts are by their nature utterly unique, meaning there are no fixed patterns that they have to follow. However, there are certain conventions that many table authors follow, so you'll start to see patterns after you've looked at a few scripts.

VP scripts are written in the Visual Basic language. (Which makes for some confusing initials: VP scripts are VB scripts!) If you want to be more technical, VP actually uses a variant of Visual Basic called "Visual Basic Scripting" or VBS. Beware example code you find on the Web, because many Web examples of "Visual Basic" use a different variant known as "Visual Basic for Applications" or VBA. VBA is much more powerful, so unfortunately, many generic Visual Basic examples on the Web just won't work in VP's simpler version of the language.

Option variables

As mentioned above, many VP table authors follow common conventions and patterns for how scripts are arranged. One of these common patterns you'll often see is a set of "option variables" defined near the top of the script, that let you select some pre-programmed variations on the table's behavior. It's always a good idea to scan through the script for a new table you've installed to see if it has any option settings and customize them to your liking.

To see if a table has any option variables, read through the comments near the top of the script. A comment in VP starts with an apostrophe ('), and the VP editor usually shows it as green text:

```
' Funhouse / IPD No. 966 / Williams, November, 1990 / 4 Players
' VP9 12.0 by JPSalas 2009
```

Script options are typically defined as Visual Basic variable assignments or Const (named constant) definitions. Most authors group these near the start of the script, to make them easy for people to find without having to read through the whole of the script, and prominently label them with comments so that you'll know what they're for.

For example, here are the options at the top of *Whirlwind* for VP 9:

```
' *****
' OPTIONS
' *****

' Controller
' 1=VPinMAME, 2=UVP backglass server, 3=B2S backglass server
Const cController          = 3

' DMD rotation
' 0 or 1 for a DMD rotation of 90°
Const cDMDRotation         = 0

' VPinMAME ROM name
' enter string of valid ROM
Const cGameName             = "whirl_13"

' flasher and GI on or off option
```

```

' 0 or 1 to disable or enable the flashers
Const Flashers_ON          = 1

' 0 or 1 to disable or enable GI
Const GI_ON                = 1

' some cabinet sound options

' 0 or 1 to disable or enable the flipper sounds
Const Flippers_Sound_ON    = 1

' 0 or 1 to disable or enable the slingshot sound
Const SlingShot_Sound_ON   = 1

' 0 or 1 to disable or enable the bumper sound
Const Bumpers_Sound_ON     = 1

' some more table options

' 0, 1 or 2 to set 'storm sound': 0 is off, 1 is fan, 2 is storm
Const StormMode            = 1

' 0 or 1 to disable or enable the "fan rotated" Williams W
Const RotatingWilliamsW_ON = 1

' 0 or 1 to choose the standard or blue colored apron
Const BlueApron_ON         = 1

' 0 or 1 to aim the plunger outlane: 0 up the inner orbit or 1 up the ramp
Const Plunger2Ramp_ON      = 1

```

You don't have to be much of a programmer to know what to do with these: just change the number after the "=" in any line where you want to change to a different setting.

How to fix up tables for a real plunger

Many VP 9 tables require some scripting changes before they'll work properly with a plunger device. Most VP 10 tables work with plungers automatically, but you might run into a few that need the same kind of fixup as is often needed for VP 9. The changes are sometimes fairly complex, so we cover this topic in a separate chapter: Chapter 39, Fixing VP Plungers.

How to enable B2S backglasses

Most VP 10 tables will work with B2S without any modification, as will some later (2016+) VP 9 tables. Earlier VP 9 tables often require some slight modifications to the table script to enable backglass art, though. See Chapter 16, Backglass Software Setup for details.

How to play table sound effects through the backbox speakers

If you have a separate set of playfield effects speakers inside your cabinet, VP decides whether to use your main backbox speakers or your playfield effects speakers as follows:

- If the sound comes from the game's ROM (the original game's software, being emulated in VPinMAME), it's played through the backbox speakers
- Otherwise, it's played through the playfield effects speakers

If you don't have a separate set of playfield effects speakers, all sounds are played through your main speakers. See "Playfield effects speakers" in Chapter 41, Audio Systems for more about setting up the extra speakers.

Assuming you do have playfield effects speakers, you might want to override the rule about playing all of the non-ROM sounds through the playfield effects speakers. VP lets you override it on an effect-by-effect basis.

First, let's think about why the rule is set up this way in the first place. The ROM soundtrack is the game's original soundtrack from the arcade game, so on the *real* version of the machine, all of the sounds from the ROM were played back through the real machine's backbox speakers. So it makes sense that we'd want to do the same thing in a virtual cab. What about the "non-ROM" sounds? Those are sound effects that the VP table author added into the simulation of the table. These are almost all meant to simulate the sound made by something mechanical on the playfield, like the ball rolling around and bumping into things, bumpers bumping, etc. So in almost all cases, you want these to sound like they're coming from the playfield area rather than from the backbox.

Now let's think about why you might want to override this for some sounds. Occasionally, you might have a mechanical sound that actually would have come from the backbox on the original real machine. For example, some EM-era machines had scoring bells situated in the backbox. Likewise, any simulated score reel sounds ought to come from the backbox area. In addition, some tables might have the occasional added voice or music effect that supplements the game's original ROM soundtrack, so you might want these to play through the backbox speakers as though they were part of the ROM soundtrack.

In VP, table sound effects are tied to one or the other set of speakers (playfield or backbox) on an effect-by-effect basis. All of the sounds are initially set to play through the playfield effects speakers. To change an effect to play through the backbox speakers instead, here's the procedure:

- Launch VP
- Open the game in the VP editor (don't run it)
- On the menu, select **Table > Sound Manager**
- Find the sound you want to redirect to the backbox speakers and select it in the list; you can use the Play button to listen to each sound if you're not sure it's the one you're looking for
- Check its current speaker assignment:
 - In VP 9, if the "Import path" looks like a regular file name, it's assigned to the playfield effects speakers; if it says ***Backglass Output***, it's assigned to the backbox speakers
 - In VP 10, the "Output" column will say either **Table** (plays through the playfield effects speakers) or **Backglass** (plays through the backbox speakers)
- If it's not already on the backbox speakers, click **Toggle BG Out** (VP 10) or **To BG Out** (VP 9)

If you want to go the other direction - change a sound that's already on the backbox speakers to use the playfield effects speakers instead - the process is exactly the

same with VP 10. Just select the sound in the list and click **Toggle BG Out** to switch it back to **Table** mode. The process in VP 9 is rather ugly: you have to export the sound effect to a WAV file and re-import it. What's more, some VP versions have a bug that won't let you export a sound that's been set to the backglass output, so you're kind of stuck. The best workaround would be to download a fresh copy of the table, export the sound from that fresh copy, and import the sound into your modified version of the game.

What about changing some of the ROM sounds to play back through the table effects speakers? Sorry; it can't be done. All of the ROM sounds are handled by VPinMAME, which doesn't have any options for changing the speakers for a specific sound. Remember that the ROM software is more of a "black box" than a VP table, since it's emulating an old arcade machine that didn't work like a PC with modern abstractions like WAV files. VPinMAME doesn't have any way to make a simple list of the sounds in a ROM that you could use to choose speakers like you can with the table sounds in VP.

How to enable DOF

DOF support is similar to B2S support: for most VP tables and some later VP 9 tables, DOF support is automatic, whereas earlier VP 9 tables usually require some script modifications. See Chapter 46, DOF Setup for details.

Removing sound effects for DOF play

If you have DOF mechanical feedback devices (solenoids, gear motors), you'll usually want to disable the digitized sound effects that tables play back to simulate the same events, since the digitized sounds tend to sound fake (not to mention redundant) when real mechanical devices are firing at the same time. Chapter 46, DOF Setup describes how to remove the unwanted sound effects.

How to fix EM tables that use the wrong coin keys

Some re-creations of EM (electro-mechanical) tables use the "wrong" keyboard keys for some functions, especially the coin-in buttons. If you're having problems with an EM game where it won't respond to your pin cab's coin buttons, this might be the cause.

The reason you see this in EM tables in particular (as opposed to more modern "solid state" games - the type with electronic displays of some kind) has to do with VPinMAME. VPinMAME is the part of the Visual Pinball system that normally handles most of the keyboard functions, including coin handling. The thing is that EM re-creations don't typically use VPinMAME, because VPM's function is to emulate the original ROM software from an electronic game. Part of the definition of "EM" is that it doesn't have any software, ergo no VPM involvement. And without VPM, it's completely up to the table script to handle all of the keyboard interaction, including the coin keys. EM table authors often hard-code the coin function to a specific keyboard key, which might not match your pin cab's button setup.

Fortunately, it's not too hard to fix these when you find them. The procedure is to find the place in the table's script where the coin key is handled, and change the script to test for the correct key.

- Open the table in the VP editor
- Open the table's script
- Search for the string "`_KeyDown`". This should take you to a line that looks like

this:

```
Sub Table_KeyDown (ByVal keycode)
```

- Note that the "Table_" prefix might be different in the actual table, but the rest should be the same. This is the start of the key handler subroutine. The code we're looking for now is somewhere in this subroutine, which is all of the code up until the next line like this:

```
End Sub
```

- Most people indent the code in this section to make it easier to see that all of the code up to the End Sub goes together.
- At this point, you'll have to read through the code to find the section that handles the coin input. Hopefully, the table author will have put in a comment, or at least used well-named variables. Look for the words "coin", "credit", or maybe something like this:

```
Credits = Credits + 1
```

- If you can find the right section, it should be preceded by a test for the key code. That will usually look like one of the following:

```
Case 6:
```

```
If KeyCode = 6 Then
```

- The number after "Case" or "KeyCode=" might be different. It's usually 6, which is the scan code for the "5" key on the keyboard (confusingly!), since that's what most desktop users expect for the coin-in key. It might also be 4 (the scan code for the "3" key), since that's another common coin-in assignment.
- If you find that line, change the number to the word AddCreditKey
- Close the script and save the table

The special symbol AddCreditKey is VP's way of referring to the key assigned to the coin function in the VP option settings. If the script was using a hard-coded scan code, this change should make the table use the correct key as set in the options.

19. Cabinet Building Tools

It's hard to overstate the importance of using the right tool for a given job. Good tools can make a seemingly difficult task easy, and can let an amateur produce professional-looking results. Here are some recommendations for the tools needed to build a virtual cab.

Basic hand tools

These core tools are needed for the most basic DIY projects. You'll probably already have most of them on hand for routine home maintenance needs. You'll probably need most of these even if you're starting with a pre-assembled cabinet, and you'll certainly need them if you're assembling a cabinet from a flat-pack kit or from scratch.

- Screwdrivers: a basic set of Phillips and flat-head screwdrivers in assorted sizes
- Hex nut driver set with assorted English and metric sizes
- Hammer
- Pliers
- Needle-nose pliers
- Sheet-metal shears
- Assorted wood and metal files

Basic power tools

Some basic wood-working power tools are good to have on hand even if you're starting with a pre-assembled cabinet, to facilitate finishing work and simple customizations.

- Drill
- Assorted drill bits, from 1/8" to 1/2"
- Hole saws for your drill, various sizes (3/4", 1", 1 1/8", 1 3/8") (some people swear by Forstner bits, but I find hole saws are easier to work with for these sizes)
- Power screwdriver (optional, but makes for less manual labor)
- Power sander (essential for surface preparation if you're going to apply paint or decals)
- Jigsaw (not essential, since a router can handle just about any job where a jigsaw would be useful, and a router can often do it better; but a jigsaw is easier for quick-and-dirty cuts)

Saws

If you're building a pin cab from scratch, you'll need one or more of the following:

- Table saw
- Track saw
- Band saw

A track saw is basically a circular saw that runs along a metal track, allowing you to

cut nice straight lines at precise locations. This is a great tool for cutting up large sheets of plywood, and for making cuts at arbitrary angles (such as the sloped side walls of a pin cab). A regular circular saw can work for this, too, but it's more difficult to cut perfectly straight lines with finish quality.

Most woodworkers would probably pick the table saw if they could have only one type of saw. Table saws are extremely versatile and are capable of precise, repeatable work. I think the ideal setup is to have *both* a table saw and a track saw. Table saws are perfect for about 90% of the cuts needed to build a pin cab, but a few of the larger pieces in a pin cab are difficult and cumbersome on a table saw. Track saws excel at handling large pieces.

A band saw can probably do everything a table saw can do. Some woodworkers consider them superior and safer tools. I don't have any experience with band saws myself, so I don't have much of an opinion, but I can appreciate their inherent safety advantages. Band saws don't tend to cause "kickback" (throwing a work piece back at the operator at high speed), which is a major hazard with table saws.

Table saw accessories: If you go with a table saw, there are a couple of accessories that are worth buying along with it.

- Finish blade for plywood. Table saws usually come packaged with coarser blades meant for cutting solid wood. Plywood is more delicate because of its layered structure, so it's better to use a finer-tooth blade. Look for a blade labeled as a "finish" blade or a "plywood" blade - these usually have 40 or more teeth and will produce a smooth edge that will need little or no sanding. The same goes for track saws, but those usually seem to come with finer-tooth blades by default.
- Push block with a "tunnel" for the blade, such as a Microjig Grr-ripper or Delmar Tools push block. These make it easier and safer to push work pieces through the table saw, especially when making narrow rip cuts (lengthwise down the board).

Router

If you're building a cabinet from scratch, it's good to have a router on hand. This is useful even if you're building from a kit, since you might want to add extra openings beyond what comes with the kit. A router is a versatile power tool with a high-speed rotating bit that can move over a piece of wood to cut grooves, holes, and edges. Some of the things a router can help you accomplish:

- Cutting custom-shaped holes
- Forming joinery edges (bevels, miters, rabbets, dados)
- Cutting grooves
- Routing out depressions or hollows

If you're starting with a flat-pack kit, most of the cuts and joinery edges should be pre-cut, but a router is still useful for a few tasks that the kits usually leave for you to do. In particular, a router is required to cut the edge grooves needed to install the plastic holders for the playfield glass, and you can also use it to cut custom holes for speakers, fans, and buttons. For this type of light usage, a hand-held router is adequate; good options are available for under \$100.

Hand routers come in two main types: fixed-base routers and plunge routers. A plunge router has a spring mechanism that lets you lower the bit straight down into the work piece while keeping the base flat against the work piece; this is useful for

routing grooves and cutting openings in the interior of a work piece. Plunge routers usually have a latch that locks in a depth, effectively making it the same as a fixed-based router, so plunge routers are the more versatile of the two types. However, plunge routers won't always fit into router tables (see below), so check compatibility before buying a router and table. Some routers come with both fixed bases and plunge bases that you swap as desired.

Recommended router bits:

- For general hole-cutting and dados, a basic set of straight bits in assorted sizes (1/4", 3/8", 1/2", 3/4").
- For installing the playfield glass guides, a slot cutter bit with a 3/32" groove width. Freude makes a suitable groove cutter with a 9/16" groove depth and 3/32" groove width (part number 63-106).
- If you're building a cabinet from scratch, there are special router bits you can use to make certain types of corner joins for the main cabinet. My preference is type of a corner join that doesn't require special bits, as described in Appendix 12, Lock Miter I: The Plywood-Friendly Way.

Router accessory:

- Circle jig for your hand router. This is an attachment for the router that lets you cut circular openings of just about any size. This is good for cutting large circular openings (larger than a drill bit can make). It's fairly easy to create a circle jig yourself (look it up on Youtube), or you can buy one.

Router table

Some routing tasks require a table-mounted router, and some are just easier with a table.

If you have a hand router, you can buy a bench-top table that you can attach your existing hand router to. There are several "universal" router tables available that will work with most brands of routers, so you probably don't need to buy a table made especially for your router - it's usually possible to mix and match brands.

Electronics

- Soldering station. If you're doing even simple electronics work, it's worth investing in a decent soldering station. A soldering station is different from a basic soldering iron in that a station has a thermostat that controls the tip temperature, which maintains consistent soldering conditions. Stations also heat up much more quickly and have much better tips than cheap soldering irons. I'm very happy with my Hakko FX88D (available for under \$100). If you've been frustrated in the past trying to do soldering work with a cheap iron, and you think it's because you don't have the right skills, you'll be amazed at your overnight transformation into a soldering genius when you switch to a proper soldering station.
- Solder. Another thing that will amaze you by improving your soldering skills overnight is to switch to a good solder. The stuff they sell at Home Depot might be okay for plumbing and other rough work, but it's not very good for electronics. The type I like is Kesler 44 63/37 Sn/Pb rosin core solder.
- Digital multimeter. An essential tool for troubleshooting electronics. The main functions I use regularly are continuity testing, voltage, resistance, and current. Virtually every meter available will have these basic functions.

One feature you should definitely look for is "auto-ranging". That means that the meter automatically senses the order of magnitude of the reading for each input type (rather than requiring you to select the range with the dial). The cheapest meters (in the \$10 range) lack auto-ranging. It's worth a few extra dollars to get this feature.

I don't have any specific brand recommendations. My professional electrical engineer friends have always sworn by their Fluke meters, and I'm sure they're great products, but they're quite pricey. You can find less prestigious brands with similar capabilities for as little as \$20. I think that even the cheap meters are pretty good at this point in terms of accuracy and features, thanks to the relentless march of progress on digital electronics, although they probably lack the build quality of the Flukes and other top brands. If you're looking for a meter for occasional hobby use, I'd buy based on price and user reviews.

20. Cabinet Parts List

When I built my cabinet, one of the unexpectedly big jobs was just figuring out what I needed to buy. This chapter is an attempt to save you some of that legwork by presenting a master list of everything that goes into a virtual pinball machine. The list is organized into categories to make it easier to digest and easier to find things.

The list starts below after a few preliminary notes.

Pinball part references

Many of the items on the list are replacement parts for real pinball machines. When possible, these are listed with the original manufacturer part numbers. This makes it easy to find the exact part you're looking for, since most arcade suppliers include these numbers in their catalogs and databases. You can enter one of these numbers into the search box on most pinball vendor Web sites to find that exact part, without having to wade through a ton of hits for similar items, as is often the case if you search by name or description. Most of the part numbers are even unique enough to yield good results from a Google search.

The part numbers listed are mostly for Williams/Bally 1990s era machines, also known as WPC machines (for "Williams Pinball Controller", the core electronics platform used throughout that generation). Those machines had a very uniform cabinet design, and most of the core cabinet parts used a single design shared across many games. Williams is no longer in the pinball business (much to the regret of pinball enthusiasts), but the modern machines being made by Stern and a few smaller boutique pinball companies still hew very closely to the WPC cabinet design, and use most of the same parts, or equivalents that can be used interchangeably. As a result, it's easy to find new replacement parts for most of the WPC cabinet components, which makes the WPC hardware an excellent basis for building a new cabinet from scratch.

Fasteners

Pinball machines use a lot of different fasteners, including machine screws, wood screws and sheet-metal screws, carriage bolts, hex nuts, flange nuts, and T-nuts. Most of these are generic parts that you can buy anywhere, but some of them are unusual and can be hard to find outside of the pinball vendors.

Here's a quick overview of the terminology and sizing specifications for most of the fasteners used in a pin cab.

Wood screws and sheet-metal screws have pointy ends that are designed to penetrate soft material and form threads in the material as you screw them in the first time. Machine screws and bolts have flat ends that can only be used with pre-threaded receptacles, usually nuts.

Machine screw sizes are specified by three quantities, like so:

diameter-thread x length

For example:

#8-32 x 1" bolt = diameter #8, thread pitch 32, length 1"

There are two unit systems for this:

- The American (also known as Imperial) system, used for most American

products, expresses the diameter in inches or "#" units (see below); it gives the length in inches; and the thread pitch is expressed in *threads per inch*. So a #8-32 x 1" machine screw has a diameter of #8 (about 5/32"), 32 threads per inch, and 1" length.

- The Metric system, used almost everywhere outside of the US, expresses everything in millimeters. Metric parts are designated with an "M" before the diameter. An M5-0.8 x 12mm machine screw has a diameter of 5mm, thread pitch of 0.8mm between threads, and a length of 12mm.

Wood screws and sheet metal screws are sized by just the diameter and length, as in #6 x 1". Since they form their own threads when screwed in, you don't need a matching part with the same thread pitch, so that's not usually specified. Sheet metal screws and wood screws are basically the same thing, with subtle differences in the way the threads are shaped. Sheet metal screws come in a bewildering variety of options for the exact shape of the pointy end, with the different shapes being optimized for tapping into particular types of material, but I've never found any virtual pinball situation where this matters. Sheet metal screws often work well with wood, and in fact, they're often used this way in the original pinball machines.

The length of a screw or bolt is the portion excluding the head.

The "#" diameter units are used in the American system for sizes less than 1/4", where the fractions become inconvenient to write. (In this context, "#" is pronounced "number". Modern audiences will be tempted to call a #8 screw a "hashtag eight", but that will get you funny looks from machinists.) Each "#" size represents an exact size in fractions of an inch, but I don't think there's any kind of formula for it; you just have to look it up in a table. Higher "#" numbers are larger diameters: #10 is bigger than #8, which is bigger than #6. Here's a quick reference to the "#" sizes commonly used in pinball machines (you can find more comprehensive tables on the Web):

# Size	Decimal Inches	Nearest fraction	Metric
#4	0.109375"	7/64"	2.778mm
#5	0.125"	1/8"	3.175mm
#6	0.140625"	9/64"	3.572mm
#8	0.15625"	5/32"	3.969mm
#10	0.1875"	3/16"	4.763mm

Every hardware manufacturer and hardware store uses these standard units to label their parts. #6 always means the same thing in a machine screw no matter where you buy it. Note, however, that M6 and #6 are different sizes, and can't be used together. This can be confusing because the "M" sizes and "#" sizes happen to look very close to each other when you eyeball them, but they're not close enough to actually mix and match parts.

For machine screws, the *thread* number (the "32" in "#10-32", for example) is an important extra spec giving the thread pitch (the number of threads per inch). Screws and nuts will only fit together if they have the same diameter *and* thread pitch; for example, a 3/8"-20 bolt won't fit into a 3/8"-32 nut, because the threads are spaced differently.

The terms "machine screw" and "bolt" are basically interchangeable. "Bolt" is usually used for larger parts, above about 1/4", but otherwise a bolt is just a big machine screw.

Here are the most common fastener types you'll see in the parts lists:

- Machine screws with slotted heads for tightening with a screwdriver (usually flat or Phillips)
- Hex-head machine screws and bolts, for tightening with a socket wrench
- Carriage bolts. These are bolts with smooth rounded heads, for places where an external bolt should be inconspicuous and not easily removable from the outside. They have square necks that fit into square holes on the receiving end, which is what serves in place of a wrench to hold them still when you're tightening a nut on the other end. Most of the carriage bolts used in a pinball machines are available with a black finish that makes them blend better with the artwork.



- Flange nuts are hex nuts with integral washers. Whiz flange nuts or flange lock nuts have serrated surfaces on the bottom of the integrated washer to help lock them in place when tightened.



- T-nuts are threaded sockets that are installed permanently in a piece of wood. These are used when you need to be able to screw a bolt into an internal location that you can't access to insert a regular nut by hand. Some T-nuts have prongs that let you pound them into like a nail to secure them, while others can be screwed in to their install location with wood screws.



SEMS screws and lock nuts

Almost all of the machine-screw fasteners in a commercial pinball machines are special vibration-resistant variations of the basic types. It's easy to understand why, given the amount of mechanical action in a physical pinball game. I think it's worth using these parts in a virtual cabinet, too, especially if you're including tactile feedback devices. It's annoying to have to keep re-tightening screws that shake loose over time, and parts that work themselves free can cause damage.

There are three special fasteners in particular that are used over and over in the commercial machines. You can substitute these just about anywhere that regular machine screws and nuts are called for.

- **SEMS machine screws** are regular machine screws with the addition of an attached lock washer at the head. The lock washer adds a lot of grip between the head and the attached part once the screw is tightened down. ("SEMS" is reportedly a shortening of "asSEMBled".)
- **Elastic Stop Nuts (ESN)s** are steel hex nuts with nylon thread inserts that add a lot of friction, requiring extra torque to turn the nut. Also known by the trade name Nyloc nuts. Note that we're not talking about nuts made entirely of plastic - the nylon part in these is just a lining inside the threading, and the main body is made of steel, zinc, or stainless steel.
- **Keps nuts** are hex nuts with permanently attached lock washers. As with SEMS screws, the lock washers add grip between the nut and the attached part when the nut is tightened. In the commercial pinballs, ESNs are much more common and could be considered the default, but Keps nuts are useful in places where you can't (or don't want to) apply a lot of torque to the screw

that the nut attaches to. ("Keps" is a trade name, taken from "shaKEProof". They're also called K-lock nuts and washer nuts.)

Cabinet trim hardware variations

The WPC cabinets were basically all the same - Williams came up with a good design and stuck to it for many years. But there was one significant variation to be aware of: a number of titles, marketed as the "Superpin" games, had extra-wide bodies for the main cabinet, allowing a wider-than-normal playfield. All of extra-wide titles came in the same extra-wide size, so even taking these into account, there are still only two widths we need to concern ourselves with: the standard machines and the widebody machines. What's more, the only thing that's different about the widebody machines is the width of the main cabinet; the other dimensions (including the backbox dimensions) and all design elements are identical between the regular and widebody machines. As a result, the widebodies share all of the same hardware with the standard machines except for the main front-top metal trim piece, known as the lockdown bar, and of course the glass cover.

If you go back further in time, before the 1990s, the cabinets become increasingly different from WPC machines. You should be aware of this when you go shopping, particularly if you shop on eBay for used parts. If you're building from scratch to the WPC plans, you'll want to make sure that any used parts you buy are compatible with WPC cabinets.

By the same token, if you're refurbishing a used cabinet from the 1980s or before, check carefully when buying new replacement parts from arcade suppliers, because arcade suppliers mostly stock parts for 1990s machines. New parts probably won't fit a 1970s cabinet unless they're specifically listed as such. If you are trying to refurbish an old cabinet, one particularly good arcade vendor to try is Marco Specialties. They have an unusually deep catalog with parts for lots of older machines. Find the original operator's manual for the machine you're restoring, if possible, since that will usually include a detailed list of the machine's parts, with manufacturer part numbers that you can look up on pinball vendor Web sites to find the exact version. If you shop on eBay, it's harder to be sure of compatibility. Ideally, look for parts for the exact title you're refurbishing, but failing that, go by manufacturer and year; the pinball makers mostly re-used parts across their product lines for a few years at a time, so a part of the same vintage from the same manufacturer will usually fit.

Where to buy

Most of the parts in the master list are fairly standardized, interchangeable parts used in most WPC-era machines, and in most cases, used in 2000s machines from Stern, Jersey Jack, and others. Most of these parts are readily available on the Web from pinball parts vendors and arcade machine dealers. If you live in a major metro area, you might even be able to find a local arcade dealer who stocks some parts, although you'll probably need to look to the Web for the more obscure stuff.

Some of the vendors I've used:

- Pinball Life
- Marco Specialties
- Planetary Pinball Supply
- VirtuaPin

Most of the generic hardware (nuts, bolts, screws) can also be found at hardware stores, Amazon, and eBay. Note that some of these are available in special finishes from the pinball vendors that you might not find at regular hardware stores (e.g., carriage bolts in black, chrome bolts for attaching the legs).

Custom-cut pieces of glass can be found locally almost anywhere from window glass stores. Check for local businesses that install and repair residential windows. Custom sizes of acrylic and other plastics can be found locally at plastics stores and some hardware stores. (If you're on the west coast, check for a local TAP Plastics.) You can also buy an uncut acrylic sheet from a hardware store and cut it to size yourself with a special plastic cutter knife, but that doesn't produce as clean a cut as you can get from a pro at a plastics store.

VirtuaPin part bundles

If you're building a cab from scratch, you can save some time on shopping (and possibly save money as well) by buying a pre-packaged parts bundle from VirtuaPin. You can find these on their Web site under "Bundle Deals". They offer two packages of particular interest to new cab builders:

VirtuaPin Cab Builder's Kit: This includes most of the standard cabinet hardware items used on typical 1990s era machines (the "Williams WPC" style). The kit comes in standard-body and wide-body versions, so choose the one matching your cabinet plans. Parts included in these kits are marked in the lists below with VP Cab Kit.

I recommend this kit. It's cheaper than buying the same parts individually, and it gets you about 80% of the way to a complete cab in terms of the accessories.

The only downside is that the kit is only available in the standard chrome/stainless steel finishes for the trim parts. That's exactly what most people want, since it's the standard look on most real machines. But some of the newer Stern machines come with a powder-coat finish on most of the metal trim, color-coordinated to complement the artwork. That's a nice upgraded look that you might want for your own build. Other metallic finishes are possible as well, such as brass. Another custom upgrade that some people want is a lock bar with a "Fire" button in the middle. That requires a special lock bar and matching "receiver", which you can't currently get with the VirtuaPin kit. If you want to choose your own finishes (see "Custom finishes" below) or include a "Fire" button on the lock bar, you're better off skipping the kit and buying everything *à la carte*, since you'd throw away most of the kit.

It's also possible to adapt the standard lockbar hardware for a Fire button, but it takes a bit of work. See Chapter 23, Cabinet Hardware Installation for details.

VirtuaPin Button Kit: This includes most of the buttons in a typical virtual cab. In the list below, we've marked the items in this kit with VP Button Kit.

I'm ambivalent on this kit. It'll save you some time, but it's less of a bargain than the cab builder's kit because it includes some buttons you probably won't use. It also lacks some that you might want to add. If you don't mind doing the extra planning and shopping, I'd skip this kit and buy buttons individually, so that you can get exactly what you want.

Custom finishes

Most of the exterior metal trim - legs, side rails, lockdown bar - is available in multiple finishes. The "standard" finishes are chrome for the legs, brushed stainless steel for the lock bar and side rails, and black powder-coat for the coin door. With

some extra work, you can get all of these parts in other finishes, such as brass, gold, or just about any powder-coat color.

The big vendors mostly just offer the standard finishes, but Marco Specialties often has a few alternatives available to match recent Stern titles. Stern typically releases a Limited Edition version of each new title, with upgraded trim, usually in a powder-coat color that complements the cabinet art. Marco usually stocks a selection of such trim, but it's hit or miss. If you're lucky, you might be able to find a full set of trim in a color featured on a past Stern LE game.

You might also be able to find trim in alternative finishes from pinball "mod" sellers. A number of small vendors sell upgrade parts, including custom-finished trim, to the pinball collector market. These guys all sell online and on eBay; try a Web search for "pinball side rail" (for example) plus the type of finish you're seeking.

If you have a specific idea for the look you want, your best option might be to buy "raw" or "unfinished" trim and find someone to apply the desired custom finish. Pinball Life sells unfinished legs, side rails, and backbox hinges, specifically as a base for custom finishes. This is a much better starting point than the standard parts, because a refinisher would have to strip the existing finish off first, which is expensive and time-consuming.

You can find services online that offer custom powder coat and metallic finishes - you ship them the parts, and they do the work and ship them back. You might also be able to find a local business that does this, if you live in a major metro area. Try looking for local shops that refinish antiques and/or auto and motorcycle parts.

Master parts list for a virtual pinball machine

Miscellaneous supplies

Item	Qty
Wire: 22-24 AWG stranded	100ft+
You'll use a surprising amount of wire to connect various parts of the machine together: buttons, lights, feedback devices. It's convenient to have a few spools of wire on hand throughout the build. You can use 22 or 24 gauge wire for practically everything, and it's cheaper (by the foot) to buy wire in large spools, so I'd pick one size and buy lots of it. If you're only installing buttons, 100ft should be adequate; if you're installing feedback devices, you'll probably want at least 200ft on hand. Buy several assorted insulation colors to make the wiring easier to trace.	
Wire: 18 AWG stranded	50ft
You'll also need some thicker wire for some of the power wires and speaker wires. I recommend 18 AWG as a good general-purpose choice for these higher power wires. 50 to 100 feet should be adequate for most pin cabs. I'd start with two 25' spools, one with white insulation and one with black.	

A good quality solder makes a surprising difference in the ease of work and the quality of your results. I really like Kester 44 rosin core solder. You can get it in 1oz tubes, but the 1lb rolls are a much better deal if you think you might do any significant amount of soldering in the future.

#6 wood screws, various lengthsLots

I found that I used an amazing number of wood screws for all sorts of random tasks. The vast majority were #6 wood screws - these are the right size for all sorts of miscellaneous jobs. Keep an ample supply on hand so that you don't have to keep running to Home Depot. Recommendation: buy 100 #6 x 1/2", 100 in 3/4", 100 in 1", 100 in 3/4", and perhaps 25 1 1/4".

NailsLots

As with #6 screws, it's convenient to have a supply of various nails on hand. You'll mostly need finishing nails rather than anything heavy-duty. I mostly used 1" and 3/4" brads, so I'd recommend buying a bunch of each.

Wood glue1 tube

If you're doing your own woodworking or building from a flat pack, you'll need a good wood glue for the joints. It's a good thing to have on hand for miscellaneous jobs even if you have a pre-assembled cab.

Epoxy1 tube

Some things are easiest to assemble or attach with a strong glue. Get a two-component epoxy (the type with two tubes of goo that you mix together just before use). I don't recommend "superglues" (cyanoacrylate glues) for most cab uses.

Cabinet wood shell

Hardwood plywood, 3/4", 4'x8' sheet1 or 2

If you're doing your own woodworking from scratch, I recommend using a furniture-quality hardwood plywood for all of the cabinet pieces. This is what they used on the real machines. The 3/4" thickness is important for making the accessories fit properly. Some people use particle board or melamine, which are cheaper, but I prefer plywood. MDO plywood (a hybrid sheet product with a plywood core and an MDF veneer) is an excellent alternative if you can find it. It combines plywood's superior strength with the perfectly smooth surface finish of MDF, so there's essentially no prep work needed for paint or decals. It's possible to make do with a single 4'x8' sheet, but it's easier with two sheets. See Appendix 9, Plywood Cutting Plans for Cabinet Construction.

Plywood, 1/2", 4'x8' sheet

1

The cabinet floor and the back wall of the backbox are typically made from 1/2" material. Most commercial machines use particle board for these parts, since they're not cosmetic. I prefer plywood since it's stronger and lighter.

Flat pack kit

1

As an alternative to raw lumber, you can buy a pre-cut flat pack kit. VirtuaPin and others sell these. A flat pack has all of the cabinet pieces cut to size and ready to assemble.

Cabinet artwork

Most cab builders opt for decals printed with custom artwork. You can create your own artwork with a computer graphics program. Decals are popular because they can make your cab look just like a real machine - done properly, they make for a very professional finish. Some people prefer a simple black paint job or natural wood finish, and some go with stenciled paint decoration for a more vintage look (like machines from the 1960s or 1970s). See Chapter 22, Cabinet Art for ideas and resources.

Custom cabinet decal set

1

A set of decals covering the visible surfaces of the cabinet and backbox.

Translite decal

1

The backbox TV's display area will necessarily be smaller than the translite, so there will be some gaps around the edges. You can use decals to fill the gaps decoratively.

DMD panel decal

1

The real machines during the late 1980s and early 1990s had printed artwork filling the DMD panel, with a custom design for each title to complement the backglass artwork. The later WPC-era machines (from about 1995 to 2000) switched to generic, matte black panels, decorated only with the manufacturer logo. I personally prefer the more ornate look of the early 1990s machines, which you can reproduce using a printed decal with your own custom artwork based on the graphics theme for your main cab. If you prefer the more neutral style of the later generic panels, you can approximate that with a simple black paint job.

Main cabinet hardware

Side rails	2
WPC style: Williams/Bally A-12359-3, 01-8993-2	VP Cab Kit
Mounting tape for side rails	80 inches
Double-sided foam tape, ¾" wide, .032" (approx) thick. This goes between the rails and the side of the cabinet. About 80" length required.	
#8-32 x 1¼" carriage bolts	2
For attaching the side rails	
#8-32 hex ESN lock nut	2
These go with the carriage bolts for attaching the side rails	
Lockdown bar	1
<ul style="list-style-type: none"> • WPC Standard: Williams/Bally D-12615, A-18240 • WPC Widebody: Williams/Bally A-16055, A-17996 • WPC Custom width: available from VirtuaPin and others • Stern standard width with center "FIRE" button: search for "premium lockbar" 	VP Cab Kit
Lockdown bar receiver	1
<ul style="list-style-type: none"> • WPC (standard, widebody, or custom): Williams/Bally part A-16673-1, A-9174-4 • Stern lockbar receiver with center "FIRE" button: Stern 500-7237-00 <p>Important: This part mates with the lockdown bar, so make sure you choose a receiver that matches the type of lockdown bar you have. The Williams WPC receiver is the same for standard, widebody, and custom widths of WPC lockbars. For other brands, check the vendor site for the compatible receiver after you select a lockbar.</p>	VP Cab Kit
Leg brackets	4
Williams/Bally 01-11400-1	VP Cab Kit
#8 x 5/8" wood screws, hex-head, slotted	32
For attaching the leg brackets to the cabinet. Williams/Bally 4108-01219-11, 4608-01081-1	
Steel legs	4
Williams/Bally A-19514	VP Cab Kit
Leg levelers with nuts	4
Williams/Bally 08-7377	VP Cab Kit

Cabinet leg protectors	4
Optional; these can help protect the cabinet decals or paint from wear around the leg joints. These weren't original equipment on WPC-era machines; Marco Specialties and Pinball Life carry several options, including felt and metal versions. The metal ones are said to be better for decals, but this is moot if you trim the decals around the leg contact area.	
Leg bolts, $\frac{3}{8}$ "-16 x 2 $\frac{3}{4}$ " or 2 $\frac{1}{2}$ "	8
The longer 2- $\frac{3}{4}$ " length is easier to work with, especially if you're using leg protectors. The type sold by pinball vendors has a chrome finish and rounded dome ("acorn") head for a nicer appearance than generic hex bolts from hardware stores. Williams/Bally 4322-01125-40	
Leg bolt nuts, $\frac{3}{8}$ "-16 thread	8
Hex nuts, $\frac{5}{8}$ " outside diameter. Williams/Bally 4422-01117-00	
$\frac{3}{8}$ "-16 T-nuts	2
Install in the "shelf" at the back of the cabinet, to mate with the wing bolts installed in the backbox floor to secure the backbox in the upright position.	
Coin door	1
The WPC style is available fully assembled with the mounting frame, coin slots, slam tilt switch, operator buttons, and wiring harness, but generally without the coin acceptor mechanisms. Williams/Bally part 09-37000-1; alternate part numbers: 09-46000, 09-96017, 09-17002-26, 09-23002-1, and 09-61000-X, 09-61000-1. <i>Available as an add-on in the VP cab kit</i>	
Coin mechanisms for coin door	1-2
Optional. You need these if you want use actual coins. One "mech" is required per coin slot (the WPC doors above have two slots).	
$\frac{1}{4}$ "-20 x 1 $\frac{1}{4}$ " carriage bolts, black	6
For attaching coin door and lockdown bar. Williams/Bally 4320-01123-20B	
<i>Note: another 6 are needed for backbox</i>	
$\frac{1}{4}$ "-20 flange locknuts	6
Williams/Bally 4420-01141-00	
<i>Note: another 6 are needed for backbox</i>	

Cashbox	1
Optional. Sits under the coin slots to collect inserted coins. You need <i>something</i> for this purpose if you plan to use coins; the standard box is well designed for the job, but it's rather large. You might prefer to improvise something more compact. The standard cashbox consists of two parts: the plastic tray (Williams part 03-7626, Stern part 545-5090-00), and a metal cover (Williams part 01-10020, Stern 535-5013-03, 535-5013-02, 535-5013-01).	
Cashbox nest bracket	1
Optional; recommended for use with a standard cashbox. Attaches to the inside front wall of the cabinet just under the coin door to keep the cashbox from sliding around. Williams part #01-6389-01.	
Cashbox lock bracket	1
Optional; recommended for use with a standard cashbox. Attaches to the short dividing wall on the cabinet floor that delineates the cashbox area at the front, to anchor the cashbox when installed. Williams/Bally part 01-10030 or 1A-3493-1.	

Playfield glass

Tempered glass sheet for playfield cover	1
<ul style="list-style-type: none"> • Standard body: 43" x 21" x 3/16", Williams A-08-7028-T • Widebody: 43" x 23¾" x 3/16", Williams A-08-7028-1 • Custom cabinet: 3/16" thick tempered glass, cut to a custom size per your plans. You should be able to order this at a local window glass shop. <p>Tip: ask the shop to omit any marking or etching certifying that it's tempered glass. Glass makers might assume that you want a certification mark, since some building codes require the marking for certain uses in home construction, such as a glass shower enclosure. You don't need any certification for use in a pinball machine, though, so you'd probably prefer to omit any such markings, to avoid visual clutter.</p>	
Playfield glass rear plastic channel	1
Standard width: Williams/Bally 03-8091 Widebody: Williams/Bally 03-8091-2	
Playfield glass side rail plastic channels	2
Williams/Bally 03-7135-1	

VP Cab Kit

VP Cab Kit

Plunger

Fully assembled: Williams/Bally B-12445-1, B-12445-6, B-12445-7. 1

You can also buy the individual parts separately if you wish to customize. Pinball Life lets you choose colors for the knob and rubber tip, but you'll have to buy *à la carte* if you want a custom knob. Note that you can buy a knobless shooter rod and epoxy on your own custom knob for a unique look. Note also that springs are available in different tensions. I'd recommend a lower tension spring for virtual pinball use since you're never going to hit an actual ball: lower tension means lower speed and less cabinet rattling. The part numbers below are Williams/Bally references as usual:

- Shooter rod: 20-9253
- Shooter housing: 21-6645-1
- Shooter housing sleeve: 03-7357
- Barrel spring ($\frac{3}{4}$ " long x $\frac{5}{8}$ " diam): 10-149
- Inner spring ($5\frac{1}{2}$ " long x $\frac{1}{2}$ " diam): 10-148-1
- E-clip ($\frac{3}{8}$ " shaft, $\frac{5}{16}$ " groove): 20-8712-37
- Washers ($\frac{25}{64}$ " x $\frac{5}{8}$ ", 16 gauge, qty 2): 4700-00051-00
- Rubber Tip: 545-5276-00

If you buy a commercial plunger kit, the plunger assembly is usually included.

Ball shooter mounting plate 1

Williams/Bally 01-3535

*Note! This **isn't** typically included in the commercial plunger kits or the "complete assemblies" sold by arcade vendors.*

#10-32 x $\frac{5}{8}$ " machine screws 3

Not typically included with commercial plunger kits or complete assemblies.

Tilt-up playfield TV mounting

Hardware for the tilt-up mechanism described in Chapter 29, Playfield TV Mounting. Only needed for that design (or a similar design).

Playfield holder bracket (left side) 1

Williams/Bally 01-8726-L-1

Playfield holder bracket (right side) Williams/Bally 01-8726-R-1	1
Pivot nut, 7/16" Williams/Bally 02-4244. The 1/2" version (Williams 02-4329) will also work.	2
Carriage bolt, 3/8"-16 x 1-3/4", black	2
Washer, 3/8" x 1" outside diameter (quantity 2)	1
Hex nut, 3/8"-16	2

Backbox hardware

Backbox hinges Williams/Bally 01-9011.2-R, 01-9011.2-L	2	VP Cab Kit
Hex pivot bushings for backbox hinges 1/2" shaft, 3/4" diameter head, 1/4" hex center, 3/8"-16 thread. Williams/Bally 02-4352	2	VP Cab Kit
Pivot bushing carriage bolts 3/8"-16 thread, 3/4" long. Williams/Bally 4322-01139-12B	2	VP Cab Kit
1/4"-20 x 1 1/4" carriage bolts, for attaching backbox hinges Williams/Bally 4320-01123-20B <i>This is in addition to the 6 needed for the coin door and lockdown bar. The VP cab kit provides 10, so you need two extra for the full set.</i>	6	VP Cab Kit
Backbox hinge backing plates These go inside the backbox to strengthen the connection points for the carriage bolts above. Williams/Bally 01-9012	2	
1/4"-20 whiz flange locknuts Williams/Bally 4420-01141-00 <i>This is in addition to the 6 needed for the coin door and lockdown bar. The VP cab kit provides 10, so you need two extra for the full set.</i>	6	VP Cab Kit
Backbox latch Williams/Bally 20-9347	1	VP Cab Kit

Backbox latch bracket Williams/Bally 01-8397	1 VP Cab Kit
Wing bolts, 3/8"-16 x 2" Install in the floor of the backbox to lock the backbox in the upright position. Williams/Bally 20-9718	2
Backbox translite lock plate assembly Keyed lock to secure the translite. Not truly necessary in home use, but a nice touch to complete the look of the real machines. Williams/Bally A-13379	1
Lower speaker panel bracket A black metal "U" channel that screws into the bottom of the backbox, to hold the DMD panel in place. The only place I've seen the original part for sale is PlanetaryPinball.com, but you can sometimes find upgraded versions in custom finishes (brass, chrome) from "mods" vendors. You can substitute a generic aluminum 5/8" x 5/8" U channel from a hardware store, cut to the required length (27 1/8" for the standard backbox width) and painted black. Williams/Bally 01-8569-1	1

Translite

Tempered glass or acrylic (Plexiglass) sheet, 18 7/8" x 27" x 1/8" thick	1
Backglass side trim, Black trim pieces that attach to the side edges of the translite. Williams/Bally 03-8228-3	2 VP Cab Kit
Backglass top trim Black trim piece that attaches to the top edge of the translite. Williams/Bally 03-8228-2	1 VP Cab Kit
Backglass lift channel Black trim piece that attaches to the bottom edge of the translite, and serves as a handle for removing it from the backbox. This fits into the speaker panel H channel. Williams/Bally 03-8229-1	1 VP Cab Kit

Speaker/DMD panel (pre-WPC-95 style)

This is the separate panel at the bottom of the backbox, below the backglass, that you see on real 1990s pinballs. Many virtual cab builders reproduce this look by using the same type of panel, since it gives the machine a modern appearance and also provides an excellent place to put the audio speakers. This

whole panel is optional, though; if you prefer the vintage look where the entire backbox is devoted to a single large backglass, you can skip the panel and use a larger backbox TV.

These parts are for the pre-WPC-95 style, used on Williams machines from about 1990 through 1995. A different style was used on later machines, with a single-piece molded plastic panel; those parts are listed below. See Chapter 31, Speaker/DMD Panel for a comparison of the two types.

Speaker panel H channel	1
Black trim piece that attaches to the top edge of the DMD panel. The "H" shape makes a channel in the top that the translite fits into, to hold the translite in place. Williams/Bally 03-8265-1	VP Cab Kit
Speaker panel latch brackets	2
These hold the speaker panel in place in the backbox. Williams/Bally 01-8535	
#8-32 x 3/8" flat-head countersunk machine screws	4
These attach the latch brackets to the speaker panel. Williams/Bally 4008-01041-06	
#8-32 x 3/8" pan-head machine screws	12
For attaching the speakers and "H" channel trim to the speaker panel. Williams/Bally 4008-01005-06, or SEMS version (with attached locking washer) 4006-01003-06	
#8 external tooth locking washer	12
For attaching the speakers and "H" channel trim to the speaker panel (required only if you're not using a SEMS screw with attached washer). Williams/Bally 4703-00008-00	
4" Speaker screen for backbox speakers	2
Use for DMD panels with 4" speaker openings. The standard type is black plastic; they're also available in metallic finishes (brass, chrome). Stern 535-8081-00, 535-8081-01	
WPC-95 speaker screen for backbox speakers	2
This is specifically designed for the molded plastic WPC-95 speaker panel, but it should also be adaptable to a 5.25" speaker opening in the older style of panel. You might have to cut holes for the speaker mounting screws to make it fit properly. Williams/Bally 04-10382-7-4	
7" Speaker screen for backbox speakers	2
Can be used for DMD panels with 5.25" speaker openings, but you have to cut it to size. Williams/Bally 03-8603-1, 03-8603-3, 01-6733.	

This is the display device that goes in the opening in the middle of the panel to display the score and other graphics. Options:

- 15-16" LCD TV, monitor, or laptop display
- PinDMD 2 or 3 (commercial "real DMD" device)
- Pin2dmd (DIY open-source LED DMD device)

Speaker/DMD panel (WPC-95 style)

These parts are for the WPC-95 style of speaker panel used on Williams machines from 1995 and after. This type of panel is made from molded plastic. Machines from 1990 through 1995 used a different style with an MDF panel and plastic facing printed with graphics; the parts for the older style are listed above. See Chapter 31, Speaker/DMD Panel for a comparison of the two types.

WPC-95 speaker panel

1

Available with embossed Williams or Bally logos in silver or gold. Williams/Bally 04-10382-7A, 04-10382-7B, 04-10374-7A, 04-10374-7G.

Bushing buttons

4

Installed in the backbox to support the speaker panel. Williams/Bally 02-5223

Speaker grill

2

Usually included with the speaker panel assembly; available separately if not. Williams/Bally 04-10382-7-4

Speaker retainer rings for 5.25" speakers

2

The speaker panel assembly might include retainer rings for 5.25" and 3" speakers. For a virtual cab you'll usually want two of the 5.25" retainers, so you might need to order one separately. Williams/Bally 04-10382-7-2

Dot matrix display shield

1

Clear plastic cover for the DMD window. Might be included with the speaker panel assembly. Williams/Bally 01-13636

PC

For the sake of making this list fairly comprehensive, here's a list of the typical PC components you'll need. This is only an outline, though; we must leave it to you to decide on specific products. See Chapter 10, Designing the PC for

guidance.

Operating System	1
Windows 7, 8, or 10 recommended. Home edition is fine.	
CPU	1
A CPU with 4 or more cores is recommended, such as a current generation Intel Core i5 series.	
Motherboard	1
Choose a motherboard based on the CPU you wish to use. Motherboards are designed to work with specific CPU types, so your choices will depend on the CPU type you plan to use.	
CPU fan	1
Most modern CPUs require a powered fan to be directly mounted on the CPU itself. Your CPU purchase might include this if you buy a boxed retail package; if not, many suitable third-party options are available.	
Graphics adapter	1
Virtual Pinball isn't as demanding as other 3D games but does require at least a mid-range graphics adapter. If cost is no object, buy a high-end gaming card. But those are usually two to three times as expensive the mid-range cards, which are fine for pinball. Get a card with at least 2GB, preferably 3G or more. You generally should use only one graphics adapter even for a 2- or 3-monitor system, as performance is usually higher with one card driving multiple monitors than with multiple cards.	
ATX power supply	1
Choose the wattage capacity according to the needs of your motherboard and graphics card.	
RAM (system memory)	1
Choose RAM chips that match the specs for your motherboard. RAM contributes to performance; more is better.	
Hard disk or SSD	1
I'd recommend using an SSD (solid-state disk) over a conventional hard disk, both for the dramatically faster boot times and for the immunity to shock and vibration.	

Case or tray	1
Optional. Some people like to use a conventional PC case, but this takes up a lot of room inside the cabinet. It's more common to mount the motherboard and other components directly to the cab wall or floor. You might also consider a conventional metal case to enclose the PC parts, or a "motherboard tray" (an open frame that holds the motherboard and helps secure the expansion cards, but doesn't enclose anything).	
Fans	2+
Most cab builders include at least two standard PC case fans to move air through the cabinet. These can be mounted on the floor and/or the rear wall.	
Disk cables	1
Your motherboard will probably come with suitable cables for connecting your hard disk, but if not, you can buy these separately.	

Power line input

Power strip	1+
Most cab builders like to be able to control power to the whole system through the PC's soft power button. You can do this with a "smart" power strip inside the cab, or you buy an ordinary power strip and make it "smart" with a contactor. A separate power strip in the backbox is useful (perhaps a small one with only 3 or 4 outlets), for plugging in the backbox TV(s) and any other devices situated there.	
12VDC contactor	1
Not needed if you buy a "smart" power strip. If you buy an ordinary power strip, though, you use a contactor to make it act like a smart one. Route the line power to all devices other than the PC (including the TVs and the feedback device power supplies) through the contactor, and control the contactor via the main PC power supply's 12V output. When the PC is on, the contactor turns on and supplies power to everything else. More on this in Chapter 11, Power Switching.	

Buttons

This is a list of the most common buttons needed for a virtual cab. Many cabs have extra buttons beyond these. See Chapter 34, Cabinet Buttons for specific products to buy and a more comprehensive list of optional buttons.

Some buttons have light bulbs inside. You can hard-wire these to be always lighted, but most people want the software to be able to control them so that

they turn on and off at appropriate times during game play. To do this, you have to treat the lights as *output* devices, meaning the lights have to be connected to a separate output controller. The button controller only handles the switch part of the button. See Chapter 44, Feedback Devices Overview for more on this.

Flipper buttons

2-4

VP Button Kit

You need two for regular flipper buttons, and another two for Magna Save buttons if desired. If you want to light the buttons, buy a transparent type; if you want to light them with variable color (RGB) lights, buy clear transparent buttons.

Flipper buttons come in two lengths: 1½" and 1¾". The VirtuaPin button kit uses the longer type to mate with their switch holders. Most real machines use the shorter length, so most of the options available from pinball vendors are only available in the shorter length. If you're looking for transparent buttons for illumination but you want the longer length so that you can use the VirtuaPin switch holders, look for part 515-7791-00.

- 1½" buttons: part number A-16883
- 1¾" buttons (not transparent): part 3A-7531
- 1¾" buttons, transparent: 515-7791-00, 3A-7531-13

Flipper button leaf switches

2-4

VP Button Kit

The flipper buttons mentioned above are just the *buttons*, without the electronic switch part. The buttons have to be paired with switches. The gold standard is **leaf switches**. Some newbie cab builders really want to use microswitches instead, since they're so much easier to find and install, but it's almost universally agreed that leaf switches are the only thing that feels right. The problem with microswitches is that they have some intentional mechanical hysteresis at the point of contact, whereas leaf switches are perfectly smooth. That's critical for flippers because it gives you greater control and better tactile feedback.

For a virtual pin cab, it's best to use low-voltage leaf switches with **gold-plated contact points**. That's the only type VirtuaPin sells, so you'll be safe if you go with theirs. If you buy from a pinball vendors like Pinball Life or Marco Specialties, make sure you get the low-voltage, gold-plated type, because the pinball parts vendors also sell a high-voltage type designed for older pinball machines. The high-voltage switches use tungsten contact points, which have higher electrical resistance than gold contacts, so they don't work as well in low-voltage logic circuits.

Flipper button leaf switch holders	2-4
Optional. VirtuaPin sells their leaf switches with plastic holders that fit over the buttons and are held in place with Palnuts (below). This makes the switch positioning and installation dead simple, but be aware that these only work with the long (1 $\frac{3}{8}$ ") buttons. If you're using the more common 1 $\frac{1}{8}$ " buttons, these won't fit. The holders might also conflict with lighting devices for the buttons, such as the LiteMite boards (below).	
If you can't use the holders, it's still fairly easy to install the leaf switches, by attaching them directly to the wall of the cabinet. So you definitely don't <i>need</i> the holders. But they're convenient if you're using compatible buttons.	
Palnuts	2-4
This screw onto the flipper button shaft on the inside of the cab to hold the button in place. You need one for each button. I prefer the nylon type, because they won't run the risk of shorting any nearby wire connections. Williams/Bally 02-3000, 20-9222, 3A-7532.	
LiteMite PCBs for flipper button lighting (optional)	2-4
These make it easy to install LEDs to illuminate transparent flipper buttons. Buy the full-color RGB type to let DOF set custom colors per game. Use one per flipper and Magna Save button.	
Start button	1
Exit button	1
Extra Ball button (optional)	1
Launch Ball button (optional)	1
Coin button (optional)	1
Main PC power button	1
Tilt bob (optional)	1

Audio system

The list below shows the basic elements needed in your audio system. There are too many options to list specific products here, so most of these are generic

descriptions. There are also several common audio system configurations, so some of this equipment only applies to certain configurations.

Quick overview: The "primary" audio system is usually a pair of speakers in the backbox plus a subwoofer in the main cabinet. Some cabinets also have a "secondary" system that places a separate set of speakers inside the main cabinet to play back mechanical playfield sound effects (ball rolling sounds, flippers, bumpers, etc). This can use two speakers, two speakers plus a subwoofer, or four speakers for "surround sound". The secondary system can even replace other tactile feedback devices, especially if you're using an "exciter" (also known as a tactile speaker or tactile subwoofer) in place of the regular speakers.

See Chapter 41, Audio Systems for more details on o the various audio system configurations, and more specific product recommendations.

Amplifier for primary (backbox) audio system	1
The standard setup needs a "2.1" channel amplifier (two stereo channels, one subwoofer channel). Most people use 12VDC car amplifiers. You can skip the amplifier if you're using powered PC speakers with their own built-in amplifier.	
Midrange speakers for backbox speaker panel (primary audio system)	2
Subwoofer for main cabinet (primary audio system)	1
Amplifier for secondary audio system	1
Optional. Only needed if you have a second audio system for mechanical playfield sounds.	
Midrange speakers for secondary "in-cabinet" audio system	1-4
Optional. Use one or two speakers for a basic system, four for a "surround" system (for placing sound effects at their proper location in the playfield area).	
Subwoofer for secondary audio system	1
Optional. Used for a 2.1 or 4.1 in-cabinet speaker system.	
Tactile subwoofer for secondary audio system	1
Optional; replaces the regular "subwoofer for secondary audio system" above. This can be used as a substitute for other tactile feedback devices, or together with them.	
7" Speaker screen for subwoofer (or larger, if you're using a larger subwoofer)	1
Williams/Bally 03-8603-1, 03-8603-3, 01-6733.	

Feedback devices

Everything in this section is optional, applying only if you want to include feedback systems on your cab. If you're not sure you want feedback at all, you can build your cabinet without it initially, and add feedback systems later as a retrofit if you change your mind. It's a fairly isolated system that can be worked into a finished cabinet, although like anything else, it's always easier and neater if you plan for it from the outset. For recommendations for specific products and parts, see Chapter 44, Feedback Devices Overview.

Output controller	1
-------------------	---

USB device that receives commands from the pinball software on the PC and switches lights, solenoids, and motors on and off.

Options:

- LedWiz
- PacLed
- Pinscape Controller
- SainSmart

Power boosters	1
----------------	---

Most output controllers can only handle devices with low power, such as LEDs. Power boosters let you connect higher power devices, such as solenoids and motors.

- Zeb's booster board
- Pinscape expansion boards
- DIY MOSFET circuit (described in Chapter 111, Pinscape Feedback Outputs)
- Relays

Secondary PC power supply	1
---------------------------	---

Most cab builders install a second ATX power supply for feedback devices, so that the main PC supply isn't affected by the extra load. PC power supplies are great for 5V and 12V devices because they have very large capacity; a cheap and low-end supply is fine for this job.

24V power supply	1
------------------	---

Some devices require higher voltages. You can add an inexpensive closed-frame 24V supply if you have any devices that need it.

Step-down converter for 6.3V	1
------------------------------	---

If you're using front-panel buttons with #555 incandescent bulbs, you can supply them using a step-down converter board to convert 12V from the PC PSU to the required 6.3V. Inexpensive converters are available on eBay to convert to selectable voltage levels. You can also find fixed-voltage 6V converters at pololu.com.

Step-down converter for shaker motor	1
A second step-down converter can be used to supply your shaker motor. You can use this to reduce the voltage level if the shaking effect is too strong at full speed.	
Step-up converter for replay knocker	1
Pinball replay knockers are built for 50V supplies; they'll run on less but will make a weaker sound. A step-up converter can be used to supply them with higher voltages if desired.	
High-output RGB LEDs for flashers	5
A set of five bright RGB lights to reproduce the bright flashing lights on pinball playfields, for a more faithful reproduction of the real thing's brightness than the video rendition can achieve. Most people use 3W RGB "star" LEDs available on eBay.	
Pinball dome lights for flashers	5
Strobe light	1-2
This supplements the RGB flashers with an extra-bright white light for strobe effects. Most people use 22-LED white car strobe lights available on eBay.	
Flipper feedback simulators	2
A device inside the cabinet to simulate the tactile "thunk" of a flipper firing. You can use a real flipper assembly for the most authentic effect, but most people use a lower-cost option such as a contactor (a large relay) or a solenoid.	
Slingshot feedback simulators	2
Another tactile "thunk" effect generator. This is the same sort of effect as the flippers or bumpers, so most people use the same types of devices for all of these, but some like to vary the devices to get different effects.	
Bumper feedback simulators	6
Another "thunk" device, usually done with the same types of devices as for flippers and slingshots.	
Shaker motor	1
A DC motor with an off-balance weight attached, to make the machine vibrate when activated for a rumble or earthquake effect.	
Gear motor	1
A DC motor with an integrated step-down gear, used for the noise the gears make. This is to simulate the sound of the motors that many real pinballs use to animate playfield elements.	

Replay knocker	1
Fan	1
Usually placed on the top of the backbox, <i>à la</i> Whirlwind.	
Beacons	1
Rotating or flashing lights like on a police car, usually placed on top of the backbox. This is another light-show element, but it's also popular because so many real pinballs have these.	
Under-cab or rear-facing RGB light strips	1
RGB light strips attached to the underside of the cabinet and/or the back of the backbox, for ambient lighting while playing.	
In-cab addressable RGB light strips	1
A different type of light strip that allows each LED to be controlled individually. These can create animated lighting effects like on a theater marquis. A controller is required (below).	
Controller for addressable RGB light strips	1

21. Cabinet Body

On the forums, virtual pinball machines are always called "pin cabs". It's appropriate that the cabinet is the defining feature of these projects, since it's what sets them apart from video pinball on an ordinary PC. This chapter is about the literal *cabinet* part of that - building the physical housing of the machine.

The bulk of this section consists of detailed plans for building a replica of a 1990s-era Williams pinball cabinet, with some slight changes to accommodate video pinball instead of a mechanical playfield. With some tweaks, you can use the plans here to build a replacement cabinet for an actual WPC-era pinball machine. The plans are complete enough to build the whole thing from scratch, starting with a couple of sheets of plywood.

Before we get to the building plans, we'll discuss the reasons behind the design of the cabinet (both the original pinball machine design and our "virtual" modifications), and some important things you should consider before finalizing your own design. Every pin cab project is unique, and I think most people will want to make at least some customizations to the generic plans I provide.

This section has a lot of background information and digressions on small (sometimes trivial) details, so it's rather lengthy. When you're ready to go out to your workshop and get building, you might prefer a more concise step-by-step list of dimensions and tasks. I've tried to provide something along those lines in a couple of appendices:

- Appendix 9, Plywood Cutting Plans for Cabinet Construction
- Appendix 10, Cabinet Construction Quick Reference

Design goals

The goal for most of us is to replicate the exterior appearance of a real pinball machine. That's what I set out to do with my own virtual cab, and I'm assuming in this section that it's your goal as well. This chapter is therefore mostly a guide to building a replica of a Williams pinball machine cabinet from the 1990s, using the same materials and parts. The plan here isn't an *exact* replica, because some modest changes are needed to make the virtual-pinball elements fit better, but most of the differences are small. For the most part, you could use the plans here to build a replacement cabinet for a real machine (and just in case you actually want to do that, I've tried to point out where and how my plans diverge from the WPC design).

Even though the WPC design is more than 30 years old now, it's still more or less the most "modern" pinball cabinet design, since most new pinball titles today are built with essentially the same cabinet plan and trim hardware as in the 1990s. Pinball is a much smaller industry today than it was then, and I don't think the manufacturers have the scale these days to do as much custom designing and machining, so they keep re-using a lot of the time-tested parts and designs from the 90s. Plus, the WPC cabinet design simply got most things right; there's not much practical reason to change it. So newer machines look very much like machines from the 1990s, as far as the basic cabinet shell goes. The only major change on the outside is the audio/video panel in the backbox, which on newer machines usually features a video screen in place of a dot matrix display, but that's not really a change to the cabinet; it's just something that drops in to the standard cabinet layout. You can build the basic WPC cabinet plan and take your pick of speaker/display panel styles.

When I was building my own virtual machine, I discovered that the design of the 1990s pinball cabinets is something of a secret art. It's not secret due to any

conspiracy of silence; it's just that there's a lot of knowledge and experience that went into the design that no one bothered to write down anywhere. Not anywhere public, anyway; I imagine that there are still a lot of old engineering diagrams and blueprints gathering dust in a basement file cabinet somewhere, but those documents aren't available to hobbyists.

Owners of the real machines can learn a lot of the design details of the original cabinets through observation, but if you don't have a real machine to take apart and examine, you pretty much have to guess. I'm fortunate to have some real machines at home, and that turned out to be a huge help for building a virtual cab. Whenever I was unclear on something, I could look at a couple of actual examples to see how they did it, and see some of the variations in different model years. I took advantage of that many times. So my goal in this section is to pass along as much of that otherwise undocumented knowledge as I can, in this one place, in an order that essentially provides a recipe for building one of these cabinets.

I know that not everyone wants their cab to look exactly like a modern commercial pinball machine. You might be basing your project on the classic Gottlieb wedge-head machines of the 1970s, for example, or you might want to create something completely novel. Even if you're aiming for a different look, though, you might find it helpful to understand how the 1990s WPC cabinet design works. Those cabinets are the result of decades of refinement. Pin cabs have to solve a number of geometry and functional constraints, and you'll face many of the same issues in creating a wholly new cabinet design. It can be helpful to see what the pros came up with after many iterations.

Build, buy, or convert?

There are three main options for creating the body of your cabinet:

- Build it yourself from scratch
- Buy a new empty pinball cabinet, pre-built or as a kit
- Convert an old real pinball machine into a virtual cabinet

If you're new to virtual pinball, the approach that might seem most appealing at first glance is to convert an old real machine. Indeed, some of the earliest virtual cabs were built as conversions from machines that would otherwise have been scrapped, and it was a popular approach in the early years. It has the advantage that it starts off from the very beginning looking exactly like a real pinball machine, and it saves you the trouble of coming up with your own cabinet design and sourcing parts.

The conversion approach has become a lot less appealing lately because of increasing prices. Almost every pinball machine is a collector's item these days; old machines command surprisingly high prices even when they're beat up, since there are plenty of people looking for restoration projects. If you're lucky enough to find a donor machine that no one wants to restore, it'll probably be in such bad cosmetic condition that it might actually be cheaper and easier to start from scratch. (You should also be prepared for some negative comments on the forums, since there's a preservationist sentiment even among the virtual pinball crowd. Many people see pinballs from past decades as works of historical significance that can't be fully or genuinely re-created once lost.)

Happily, the "build" and "buy new" alternatives are both quite practical, and I consider both of them superior to conversion, even without the price considerations. You can buy high-quality reproduction cabinets that look exactly like the real thing, in kit form or fully assembled. Or, if you have some basic woodworking skills, you can

build an excellent reproduction cabinet yourself. The real machines use fairly simple designs that you don't have to be a master carpenter to reproduce. This section provides ready-made plans, so there's no research required.

What's more, it's easy to obtain all of the genuine cabinet hardware (metal rails, legs, etc) needed to fit out a custom-built cabinet and make it look exactly like a real machine. All of the hardware is standardized across machines, and several big online pinball suppliers sell the parts. You can thank the collectors for that - they buy these parts to repair and restore their machines, so there's a healthy market in the parts that keeps them readily available.

Here are my recommendations:

- For most people, I recommend using a VirtuaPin flat-pack kit. I used this approach myself, and I'm very happy with the results. It's not the cheapest option, but it's reasonably priced, and it yields an excellent finished product. It's about the same level of difficulty as assembling an Ikea product. With a little care, the result will be essentially indistinguishable from a brand new commercial pinball machine cabinet.
- If you enjoy woodworking and have a decently equipped workshop, consider a scratch build. Doing it yourself is much cheaper than buying a VirtuaPin flat pack, assuming you already have the tools, and you should be able to get equally professional results with a little care. Pinball cabinets are relatively simple as woodworking projects go; they're built out of ordinary plywood, and they're all straight lines and (mostly) easy joinery. But it does require some tools beyond the home-handyman basics, and it's a fair amount of work, so don't choose this option just because of the potential cost savings - choose it because you enjoy this sort of construction project.
- If you don't want to do any woodworking, and you don't even want to assemble a kit, you can order a fully built cabinet from VirtuaPin or from a number of other vendors. VirtuaPin's pre-assembled product is the best one I've seen, and it's the only one I know of that faithfully reproduces the design of the real pinball machines. (It's good enough that some collectors restoring real machines buy this kit to use as their replacement cabinets.) The products I've seen from other vendors use idiosyncratic designs and non-standard trim hardware, so they don't look quite like the real thing. If a realistic appearance is important to you, take a close look before buying to make sure you like the design.
- I generally don't recommend trying to re-purpose an old real pinball as a virtual cab, in part because old cabinets tend to be so beat up that restoration would be more labor-intensive than building a new one, and in part because the economics rarely pan out. More on that below.

Economics of new vs used

If you're set on the idea of re-purposing a used cabinet, I'd suggest doing a little research first to make sure you don't overpay. The question you want to ask is: would I actually save money buying the used cab, or would it be cheaper to buy the same parts new?

To answer this, ask the seller for a list of all the hardware parts that the used cabinet includes - the legs, side rails, lockbar, etc. Don't assume that everything is included, because a lot of eBay sellers strip all of the parts out and sell them separately; a used cabinet might not come with anything beyond the wood box. And only consider the hardware that you'll actually use on your virtual cab, since those are the only parts you'd have to buy if you were starting from scratch. Only count parts that you

actually need in the virtual cab (e.g., don't count the playfield, bumper caps, etc), and only count used parts if they're in usable condition.

To help you get started here's a list of the main parts that real machines and virtual cabs have in common. See Chapter 20, Cabinet Parts List for a more detailed parts list with descriptions. We left the price column blank, since prices obviously vary over time and from one vendor to the next, so you'll have to fill that in by checking current prices at your preferred vendor(s) (such as VirtuaPin, PinballLife, or Marco Specialities). For the "wood body" line item, you can use VirtuaPin's flat pack or unfinished cab body offerings for comparison. Remember, only include the parts that the seller is including with the new cab, since you want to compare new-vs-used for what you're actually getting from the seller.

Add up the prices of the new parts, and compare the result to the seller's asking price for the used cab. If the asking price for the used cab is cheaper than the new parts, and everything's in good enough shape that you can actually use it, you've found a good bargain. If the seller is asking more for a beat-up used cab than what you'd pay new, I'd pass on the deal.

<input checked="" type="checkbox"/>	Description	Price New
<input type="checkbox"/>	Main cabinet wood body	\$
<input type="checkbox"/>	Backbox wood body	\$
<input type="checkbox"/>	Legs (qty 4)	\$
<input type="checkbox"/>	Leg levelers ("feet") (qty 4)	\$
<input type="checkbox"/>	Leg brackets (qty 4)	\$
<input type="checkbox"/>	#8 x 5/8" wood screws for leg brackets (Williams ref 4108-01219-11, 4608-01081-11), or #10 screws if preferred (qty 32)	\$
<input type="checkbox"/>	Leg bolts ($\frac{3}{8}$ "-16 x $2\frac{3}{4}$ " or $2\frac{1}{2}$ ") (qty 8)	\$
<input type="checkbox"/>	Side rails (qty 2)	\$
<input type="checkbox"/>	Lockbar	\$
<input type="checkbox"/>	Lockbar receiver	\$
<input type="checkbox"/>	Coin door	\$
<input type="checkbox"/>	Coin acceptors ("coin mechs")	\$
<input type="checkbox"/>	Cashbox tray (Williams ref 03-7626)	\$
<input type="checkbox"/>	Cashbox lid (Williams ref 01-10020)	\$
<input type="checkbox"/>	Cashbox nest bracket (Williams ref 01-6389-01)	\$
<input type="checkbox"/>	Cashbox lock bracket (Williams ref 01-10030)	\$
<input type="checkbox"/>	Carriage bolts, black, $\frac{1}{4}$ "-20 x $1\frac{1}{4}$ " (qty 6: 4 for coin door + 2 for lockbar)	\$

<input type="checkbox"/>	Flange locknuts, ¼"-20 (qty 6: 4 for coin door + 2 for lockbar)	\$
<input type="checkbox"/>	Top glass	\$
<input type="checkbox"/>	Rear plastic channel for glass	\$
<input type="checkbox"/>	Side rail plastic channels for glass (qty 2)	\$
<input type="checkbox"/>	Plunger (ball shooter) assembly	\$
<input type="checkbox"/>	Ball shooter mounting plate (Williams ref 01-3535)	\$
<input type="checkbox"/>	#10-32 x ¾" bolts for mounting plunger assembly (qty 3)	\$
<input type="checkbox"/>	Backbox hinges (qty 2)	\$
<input type="checkbox"/>	Backbox hinge backing plates (qty 2)	\$
<input type="checkbox"/>	Carriage bolts, ¼"-20 x 1¼" (qty 6, for backbox hinges)	\$
<input type="checkbox"/>	Flange locknuts, ¼"-20 (qty 6, for backbox hinges)	\$
<input type="checkbox"/>	Pivot bushing carriage bolts (qty 2)	\$
<input type="checkbox"/>	Hex pivot bushings (qty 2)	\$
<input type="checkbox"/>	Backbox latch	\$
<input type="checkbox"/>	Backbox latch bracket	\$
<input type="checkbox"/>	Backbox lock plate assembly	\$
<input type="checkbox"/>	U-channel, metal, ⅝" x ⅝" x 27⅛" (backbox speaker panel holder)	\$

Where to find used machines

Your best bet for finding a used machine at a good price will be local sellers. Search your local craigslist and local newspaper classified ads. A particularly good place to find a deal is at an estate sale. Heirs often want to clear out the house quickly and won't have any sentimental attachment to an old pinball.

If you can't find anything locally, eBay will give you access to sellers nationally (and even internationally). But I wouldn't get my hopes up; it's hard to find a good deal on a used cab on eBay these days. For one thing, shipping a cabinet is expensive due to the size and weight; shipping can add about \$400 to the price. For another, eBay sellers know they can get top dollar for used cabs, so the base price is unlikely to be a bargain. Most eBay sellers are also savvy enough to strip a machine of all of the parts and sell them off separately, which largely defeats the purpose of reusing an old machine.

Kits and pre-built cabinets

If you can't find or don't want to use a salvage machine, but you also don't want to build everything from scratch, there are several companies that will sell you a brand new cabinet. For options, search the web for "new pinball cabinet" or "pinball cabinet restoration". One vendor I can recommend from personal experience is VirtuaPin. They sell cabinets in both kit form and fully assembled, and can customize them to your specifications.

You should be able to find options ranging from "flat pack" kits that you assemble yourself, to fully assembled cabinets with all of the hardware and artwork installed.

The cheapest and most DIY option is a flat pack kit. This is like an Ikea bookshelf: it consists of the wood parts, pre-cut and pre-drilled, ready for you to assemble. This is the cheapest kit option, since you provide the assembly labor, and because shipping is cheaper than for a bulky assembled cab. The degree of difficulty is slightly higher than for assembling Ikea furniture, but only slightly; no real woodworking skills are required, and you'll just need basic tools like screwdrivers and hammers. You might also have to do some sanding to even out corners and edges. And you'll have to do your own finish work (painting, staining, or applying decals). I used the VirtuaPin flat-pack kit for my own cabinet, and I highly recommend it. They use a good furniture-grade plywood, and the design is a faithful reproduction of the WPC cabinets that Williams shipped in the 1990s, so the result is exactly like a brand new real machine.

The next step up in price and completeness is an assembled cabinet shell. This is just the wood shell, typically unfinished (ready for you to paint, stain, or apply decals), and without any of the cabinet hardware accessories installed. This eliminates the assembly work required for a flat pack. It's considerably more expensive to ship because it's so bulky.

At the high end price-wise, you can buy a fully assembled cabinet with all of the hardware and graphics pre-installed. VirtuaPin sells these in addition to their flat-pack and assembled-but-unfinished products. All of the VirtuaPin options use the same materials and design as their flat pack, so they all yield excellent reproductions of the 1990s Williams machines. There are other vendors selling pre-built cabinets as well, but check their designs carefully before buying, because I've seen a couple of other vendors who use their own ad hoc designs that look a bit cheap and cheesy to my eye.



Tip: Ask about the button hole layout. If you order a kit or pre-built cabinet, ask the seller for details on the locations of button holes they pre-drill, and ask them to customize the drilling to your plans if you have something else in mind. The vendor might drill holes by default that you don't want. Pay particular attention to the placement of the plunger and flipper button holes. Many virtual cab builders choose non-standard locations for these to accommodate the playfield TV (see "The dreaded plunger space conflict" in Chapter 29, Playfield TV Mounting and "Positioning the plunger" in Chapter 37, Plunger). If you're not sure how you want to handle this at the time of your order, you can simply ask the vendor not to drill any holes for the plunger or other controls. That will give you the flexibility to drill them yourself later when you know how everything will fit together.

Scratch build

A scratch build is the cheapest option if you already have most of the tools required. The raw materials consist mostly of a couple of sheets of plywood. If you don't already have the tools, it's probably cheaper to buy a pre-built cabinet, but not by a huge margin - you don't need a huge set of tools. A scratch build also lets you build exactly what you want.

Lumber:

- 3/4" (nominal) plywood, two 4'x8' sheets, for almost all of the main cabinet and backbox. Choose a quality hardwood plywood that's graded for furniture or cabinetry use. Commercial pin cabs are typically birch. The 3/4" thickness is important, as many of the trim parts and controls (like flipper buttons and plunger) are designed to fit into 3/4" walls.

MDO plywood is even nicer, if you can find it. MDO is a hybrid product with a plywood core and an MDF veneer, so it has the strength and lightness of plywood and the flat, grainless surface finish of MDF. The MDF veneer saves hours of prep work for painting and decals.

It's just barely possible to fit a standard-width cab into a single 4' x 8' sheet, but it's much easier with two sheets. See Appendix 9, Plywood Cutting Plans for Cabinet Construction for layout suggestions.

- 1/2" (nominal) plywood or particle board, one 4'x8' sheet, for the cabinet floor and the back wall of the backbox. Most commercial machines use particle board, to cut costs, since these pieces aren't cosmetic. Plywood is a nice upgrade if your budget permits. It's stronger and lighter.
- A length of 2x2 (nominal) board, for corner braces. A nominal 2x2 is actually 1.5" on a side. It usually comes in 8-foot lengths. You need at least 5 feet, so one 8-foot board will leave you with some spare material for test cuts and do-overs.
- A length of 1x2 (nominal) board, for a trim piece for the backbox. You can also fashion this out of the same 2x2 as the leg braces, but it's easier with a 1x2. This board needs to be at least 28" long.

"Nominal" lumber dimensions refer to what the lumber yard calls it, not the true size. Plywood is generally 1/32" thinner than the nominal thickness, and dimensional lumber, such as 1x2 or 2x2 strips, is generally 1/4" to 1/2" less in each dimension than the nominal size. So when the list above calls for 3/4" plywood, it means you should buy what the lumber yard calls 3/4" plywood, even though the true thickness will be slightly less than that.

Some cab builders use MDF (fiberboard) instead of plywood for the whole build. I prefer plywood, but MDF works too. Each material has some advantages over the other. MDF is cheaper, and it's nearly perfectly uniform in thickness and composition. MDF sheets also tend to be perfectly flat, whereas plywood often has some warping, even fresh from the factory. The downsides of MDF are that it's heavier, it's not as sturdy or as durable as plywood, it doesn't hold screws as well, and it can sag over time.

Most commercial cabinets use a mix of plywood and particle board: plywood for the cab and backbox walls, and particle board for the cabinet floor and the back of the backbox. I'm normally in favor of faithfully replicating the original details, but in this case I see no benefit, since the particle board in the originals was purely for cost reduction and not for any functional reason. It's a worthwhile upgrade to use plywood for everything.

Hardware: See Chapter 20, Cabinet Parts List.

I personally prefer to use real pinball parts wherever possible, instead of trying to improvise something out of common hardware parts. It can be a bit more expensive to use the real parts, but they tend to look better, and in many cases they're easier to work with because they're purpose-built for a specialized job. An exception is that many of the fasteners and generic nuts and bolts that you can buy anywhere. But even some of the generic-sounding hardware can be hard to find outside of pinball vendors. Anything that I listed with a Williams part number in the master list (Chapter 20, Cabinet Parts List) is probably hard to find outside of pinball suppliers.

Tools: See Chapter 19, Cabinet Building Tools for recommendations on the woodworking tools you'll need for building the cabinet.

Choosing a cabinet design

As far as I'm concerned, there's only one cabinet design that we need to concern ourselves with: the WPC cabinet. This is the cabinet that Williams (and its co-brands Bally and Midway) used for nearly all of their machines made during the 1990s, which were based on the electronics platform known as WPC, for Williams Pinball Controller. Thus the name "WPC cab". I don't know if Williams ever had an official internal name for this series of cabinets, but probably not; I don't think they thought of it like that. My guess is that they treated each game's cabinet as unique, as suggested by the unique part number assigned to each game's cab. So "WPC cabinet" is a made-up term of convenience, not an official name - and it's also not quite precise, in that the last dozen or so System 11 titles also used the same design. The wood shop apparently didn't coordinate their updates in lock-step with the electronics department. But even so, "WPC cab" is a pretty good name for our purposes, and other pinball people will probably know what you're talking about if you refer to it.

One good reason to use the WPC cabinet design is that it's by far the easiest to find parts for. Machines from this generation are still widely deployed, and they share a lot of the same parts (Williams quite intentionally re-used parts across titles to keep their own costs down), so there's plenty of demand for most of the common cabinet parts even today. If you design to the WPC specs, you'll be able to take advantage of that, since your machine will be compatible with the same readily-available parts. Using real pinball parts gives your machine an authentic look, and it's a lot easier than engineering and fabricating your own custom metal parts.

Another reason to use the WPC cabinet plan is that it still looks up-to-date, because most newer commercial machines are still using the same exterior design. Stern's latest machines still look almost identical, as do the machines from the boutique pinball makers like Jersey Jack and Spooky Pinball. This cabinet style is what you'll see almost every time you encounter a recent pinball title in an arcade or bar. A virtual cab following the same plan will look exactly like what everyone expects a real pinball machine to look like.

For a DIY project, you're always free to come up with something completely different, either to fit your particular needs or purely to be unique. If you have other ideas for how your cabinet should look, you can take as much or as little from the WPC design as suits you.

We provide detailed plans for the WPC design later in this section. Before we get to the plans, though, there are some variations that you might want to consider, so that you can customize the plans for your project's specific goals.

Standard and Widebody cabinets

The WPC design is what we usually call the "standard" cabinet, because Williams/Bally/Midway used this for most of the machines they shipped in the 1990s. However, they also used a variation of the plan with a slightly wider main cabinet, to accommodate a larger playfield. This wider variation was used for seven titles in all: *The Twilight Zone* (1993), *Indiana Jones: The Pinball Adventure* (1993), *Judge Dredd* (1993), *Star Trek: The Next Generation* (1993), *Popeye Saves the Earth* (1994), *Demolition Man* (1994), and *Red & Ted's Road Show* (1994). The official marketing name for these games was "Superpin", but no one ever uses that term; everyone in the virtual pinball world calls these the "widebody" machines.

The widebody design is identical to the standard WPC cabinet in almost every detail, except that the main cabinet is 2¾ inches wider than the standard body. All of the other dimensions are exactly the same as in the standard body.

A lot of people assume that the widebody WPC cabinet design had a corresponding "widebody backbox", but not so. Williams used the same backbox dimensions for their standard and widebody machines. In fact, my guess is that they chose the widebody width they did precisely because it's as wide as you can go without making the backbox wider. That doesn't mean that you can't customize the backbox size if you're using a widebody design, just that you don't necessarily have to. If you're going wider than wide, though - if your main cabinet outer width goes over about 25" - then you will have to widen the backbox, assuming you plan to use the WPC-style hinges. The backbox has to be at least 3.5" wider than the cabinet for the WPC-style hinges to work.

Widebody machines require a wider lockbar and cover glass. Pinball vendors sell widebody versions of both that fit the Superpin cabinet dimensions, so you can still use off-the-shelf parts with this design. The widebody lockbar fits the standard lockbar receiver (the hardware piece that mounts in the cabinet to hold the lockbar in place), so you don't need a special version of the receiver. All of the other hardware is interchangeable with the standard body machines.

Note that my plans in this section peg the standard-body cabinet exterior width at 21-7/8", which is what I read on the tape when I carefully measure my own original Williams cabinets, whereas most other Internet plans I've seen say it's 22". Maybe everyone else is rounding up, or maybe my cabs have shrunk a little bit over the years (they did all leave the factory more than 30 years ago). It's a pretty trivial difference, but I thought I'd mention it. The standard lockbar will fit in either case, but it might be a little snug on a 22" wide cabinet.

Custom width

In addition to the WPC standard-body and Superpin widebody designs, there's a third option: you can design a cabinet with a custom width that doesn't match either of the Williams WPC-era designs.

The main reasons to build to a custom width are to get an exact fit for your playfield TV, or to accommodate a TV that's even wider than what will fit in a WPC widebody cabinet.

A custom width requires a custom lockbar and custom glass cover. As long as you use the standard cabinet length, you can still use all of the other off-the-shelf hardware.

To order a custom lockbar, contact VirtuaPin. They're the only current source I know of, so hopefully they'll keep selling them. Expect to pay about double the price of the standard lockbar (which I think is a pretty reasonable premium for a made-to-order metal part that has to look nice).

The lockbar is mated with a second part known as the "lockbar receiver". The receiver does *not* need to be customized for different widths; the standard receiver will work with any lockbar width, as long as your cabinet is wide enough to accommodate it (a minimum of 19½ inches inside width) . Keep the minimum size in mind if you're designing a mini-cab that's narrower than the standard-body cabinets.

Pinball vendors don't sell custom glass, but you should be able to order it from a local window glass dealer. Any glass shop should be able to fabricate a custom glass sheet for you in almost any desired size. Once you know the inside width of your cabinet, order a tempered glass sheet in the required width, by 43" length, by 3/16" thickness.

When calculating the required width of your glass, take into account the overhang beyond the inside dimensions. The easy rule of thumb is to make the glass ½"

wider than the **inside width** of your cabinet (that is, the distance between the insides of the side walls). For example, the standard body cabinet has an inside wall-to-wall width of 20½ inches, so the standard playfield glass is 21" wide.

One more tip about ordering custom glass: ask the vendor to omit any marking that identifies the glass as tempered. Glass shops sometimes include a certification marking on tempered glass, in case you're planning to use it for something like a shower enclosure where tempered glass is legally required by building codes. You don't need any such marking for legal purposes in a virtual cab, so you'll probably prefer to avoid the visual clutter.

How to choose a cabinet width

If it weren't for the constraint of fitting a TV, I'd recommend the standard-body plan and leave it at that. Using the standard dimensions produces a machine with exactly the right proportions to look authentic, and it lets you use readily available off-the-shelf parts for all of the hardware. It's the easiest, most cost-effective approach, and it looks good.

But sadly, TV manufacturers don't always cooperate with our virtual pinball plans. TV manufacturers only make TVs in certain sizes, so we're stuck with whatever sizes are on offer. The TV you pick based on price and performance might not be available in exactly the right size for a standard cabinet.

What size TVs will the standard cabinet sizes accommodate? There's no absolute rule here, since the nominal diagonal size of a TV doesn't tell you the exact exterior dimensions - the only way to be sure is to measure the TV. You might also be able to find the dimensions listed in the specs on the manufacturer's Web site or on a retailer site. Generally speaking, a standard-body cab will accommodate most TVs up to 39" diagonal, and you might be able to squeeze in some 40" models; and a widebody should handle most TVs up to 45".

When I built my own cab, I chose standard-body dimensions, mostly because I wanted my virtual cab to blend in with my small collection of real standard-body pinballs. At the time, 39" TVs were readily available, so you could easily find a TV to fit the standard-body dimensions. As of 2022, though, 39" TVs are rare. The most common TV size that's in range for a pin cab today is probably 42", and that requires a widebody cabinet. So if I were building a new cab right now, I'd probably have to go with a widebody design. In any case, I'd try to use one of the original WPC sizes - standard or widebody - rather than building to a custom size tailored to the TV, so that I could use off-the-shelf cabinet hardware.

Custom length

As long as we're on the subject of custom widths, we should consider lengths as well.

As with a custom width, the main reason to build to a custom length is make a TV fit exactly. As discussed in Chapter 7, Selecting a Playfield TV, a real pinball playfield is much more elongated than a 16:9 TV screen; a typical 1990s era playfield is more like 20:9. So placing a 16:9 TV in a standard-sized cabinet leaves a few inches of dead space at the front and/or rear of the cabinet. Some cab builders don't like that idea because they want to fill every square inch with TV display area.





One way to deal with the extra space is to remove it by shortening the cabinet length to exactly fit the TV.

In my opinion, it's better to stick with the standard cabinet length and accept that there will be some extra front-to-back space. The main problem with a custom length is that you won't be able to find side rails or glass guides; no one sells those in custom sizes as far as I know, and they'd be difficult to fabricate yourself unless you have some good metal-working tools. Besides, the extra space can be put to good use for features that you might want anyway:

- If you're going to install a plunger, you might need 3-4" of extra space between the front of the TV and the front of the cab to make room for the plunger mechanism.
- Even if you don't need the space for a plunger, I still like setting the TV back a few inches from the front for the sake of sight-lines, so that you don't have to look straight down to see the flippers.
- Space at the front of the TV can be used for an "apron" similar to that on a real machine, with printed instruction cards, or even small monitors that display live instruction cards, scores, etc.
- Space at the back of the TV can be used for a flasher panel or LED matrix.

Custom backbox sizes

As with the main cabinet dimensions, some virtual cab builders choose to deviate from the standard backbox sizing to better fit a selected TV.

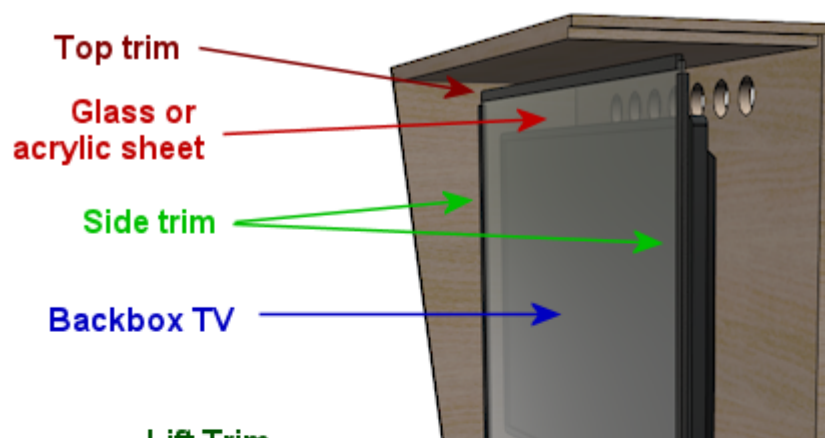
Finding a backbox TV that fits the available space can be even more vexing than finding a playfield TV. There are two big problems here. The first is proportions: modern TVs all use the 16:9 aspect ratio, but the translites in the WPC design are much squarer: approximately 13:9, closer to the old-fashioned NTSC 4:3 ratio. The second problem is that there are very few models available that are even close to the right size. The most popular current TV size that's close to what we need is probably 32", but a 32" TV is too wide for a standard backbox. The next size down tends to be 27" or 28". A 28" TV will leave about an inch of dead space on each side, and a few inches above and below. A 29" would be close to perfect, but that's never been a common size.

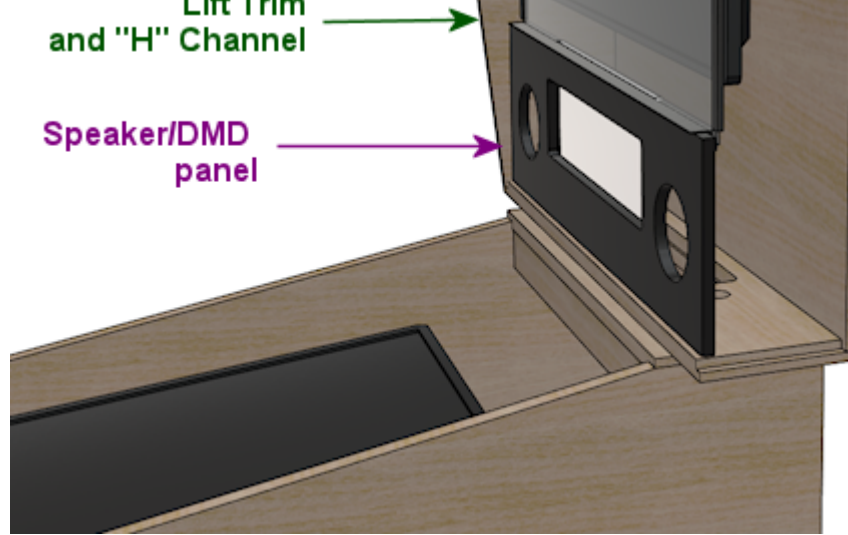
You can't control what sizes the TV manufacturers make available, so you can either live with a little dead space around the perimeter, or you can resize the backbox to fit the TV. I personally think the dead space is the better compromise, in part because it

lets you use standard off-the-shelf pinball parts, but mostly because the backbox shape will look "wrong" if you change it from the standard. This is a case where I think a lot of cabinet builders tend to fixate on the wrong thing during the planning stages, by focusing on the dead space rather than the overall proportions. The thing that you might not appreciate during the planning stage is that any dead space tends to disappear into the background when you're actually playing. Your eye sees what's there (the backglass graphics), not what's missing (the dead space around the edges). The proportions of the overall outline, on the other hand, are always noticeable.

If you do want to consider custom dimensions for the backbox, keep in mind that some of the associated hardware parts are sized for the standard dimensions, so you might not be able to use off-the-shelf parts for everything. Here are the parts that will be affected (see the illustration below if you're not familiar with all of these):

- The glass or plexiglass cover for the TV. This isn't a required part, but I recommend including it because it creates a more authentic appearance. There's no downside to a custom size for this part, because it doesn't come in an off-the-shelf version to begin with; you'll have to custom-order it even if you're using the standard size. You can have this made at any local window glass shop or plastics store, and they'll be able to cut it to whatever size you need.
- Trim pieces for the glass/plexi cover. Real pinball machines use black plastic trim around the edges of the translite. These are only available in the standard sizes. You can fairly easily cut them to smaller lengths if necessary, but there's no good way to make them longer, so you'll have to live with some gaps if you use a wider or taller than normal size.
- Speaker panel "H" channel and translite lift trim. These are plastic trim pieces that go at the top of the speaker panel and bottom of the translite, respectively, and they're sized to the standard width. If you use a wider-than-normal width, you can use the standard pieces, but there will be some gaps at the edges.
- Speaker panel. If you use a non-standard width, you'll need a custom speaker panel, assuming you're using the three-monitor configuration. No one sells those in custom sizes, so you'll have to fabricate one yourself. The ready-made speaker panels are made from plywood or particle board, so building your own only requires woodworking tools. Be aware that it's a fairly advanced project requiring precision work. You'll have to cut two large circular holes the speakers and a large rectangular opening for DMD. I'd recommend using a CNC machine (a computer-controlled cutting machine that cuts according to a digital plan). There are online services for this, such as SendCutSend. If you live in or near a major city, you might also be able to find a local CNC service, or a "maker" facility that lets you use their equipment for an hourly fee.





Mini cabs

A popular variation on the basic cab design is to scale things down a bit from the real machines. This can be especially attractive if you don't have a lot of space, and might help gain acceptance from skeptical spouses or housemates.

There's no "standard" mini-cab design, but you can find ideas from other people's builds by searching the cab forums at sites like [vpforums](#). Many people who've built their own cabs post build logs with details of their design.

If you want to design a mini-cab from scratch, you can start with the basic WPC design, and just scale down all of the dimensions based on the playfield monitor you choose. A 32" TV makes a good core to build a mini-cab around; if you scale everything down proportionally, it yields a cab that's about 3/4 of the full size. That's enough of a reduction to fit more comfortably into a residential setting, but it's still big enough to be free-standing.

A few people on the forum have shrunk things down even further, to table-top or hand-held size, using a small computer monitor or tablet as the playfield.

For a mini-cab in the 3/4 scale range, you should be able to build it pretty much the same way that you'd build a full-size cabinet. You'll have to make the same adjustments to cabinet hardware discussed above under "custom width" and "custom length", but otherwise you should be able to use standard materials (such as 3/4" plywood for the enclosure) and many of the standard hardware parts. One thing to keep in mind is that interior space will be a bit tight for the electronics, but you should be able to fit the necessary computer parts and a basic set of feedback devices.

If you reduce the scale to table-top or hand-held dimensions, you'll have to invent a lot more of the design on your own, since most of the standard hardware will be too large. That's beyond the scope of this guide, but you should be able to find one or two examples in the forums or elsewhere on the Web if you're looking for inspiration. Note also that all of the pinball software discussed in this guide is for Windows PCs, so if you're considering something else (like a tablet or Raspberry Pi) as the computer core, you'll also have to find other software to use. There are some decent commercial pinball games for tablets that could serve, but the commercial games don't tend to have any integration with cabinet features, so it might be challenging to make everything work the way you want it to.

WPC cabinet plans

We now present our WPC standard-body cabinet plans. These are based primarily on measurements taken from actual WPC pinball machines, with some additions and modifications to accommodate the peculiarities of virtual pinball. I've tried to identify all of the deviations from the real machines, for those with a special interest in accurately re-creating the originals.

Other Internet plans

There are several other pinball cabinet plans available on the Web, including other replica WPC designs. Some of the other WPC plans I've seen have slight variations from mine, so you might want to compare and contrast any others you find as a sanity check, and to see if there's anything you prefer in the variations. I've taken a great deal of care to check my plans against actual WPC machines, and I believe the version presented here is the closest to the real thing that I've seen, but of course that doesn't mean they're the ideal plans for every build, just that they're close to what Williams actually did build. You might have good reasons to deviate from that. Most of the details can be changed in small ways without much affecting the usability of the finished machine. (One detail that you probably shouldn't tinker with is the placement of the flipper buttons, since that's such a crucial part of the feel, and it's highly consistent on the real machines.)

One set of plans I'll call out in particular is Jonas Kello's Sketchup model, available on github:

github.com/jonaskello/wpc-cabinet

The nice thing about his 3D model is that you can look at it from all angles, which might be helpful whenever my illustrations leave something unclear about the spatial relationships between components. Jonas's model appears to have been prepared with excellent attention to detail. One warning, however: he explains that he took his measurements from a widebody WPC machine (*Star Trek: The Next Generation*) and adjusted them to infer the standard-body dimensions. This creates an opportunity for errors and inconsistencies to creep in, and I have in fact found a couple of errors in his model that are likely due to this. My measurements were taken directly from standard-body machines (*Theatre of Magic* and *Medieval Madness*), so even though my figures undoubtedly have inaccuracies of their own, they're at least free of that particular source of error. In addition, even where our measurements essentially agree, there are a number of slight differences, on the order of 1/16" to 1/8", which I attribute to some combination of measurement error and actual variations in the machines we sampled.

Another set of plans worth mentioning can be found in this Pinside thread by Swinks, which has measurements from original WPC standard-body machines (per the thread, mostly taken from *Creature from the Black Lagoon*, with some corroboration against *Bram Stoker's Dracula* and Stern's *Iron Man*):

pinside.com/pinball/forum/topic/bally-wms-cabinet-designs-help-needed

Finally, Greg Butcher, a/k/a mameman, drew up a set of WPC widebody plans many years ago that's often referenced in the virtual cab forums. They have some inaccuracies in the details of the construction, but they're still a useful reference. Jonas Kello captured them in his github repository:

github.com/jonaskello/wpc-cabinet/blob/master/references/williams%20widebody%20cabinet%20rev3.pdf

Joinery

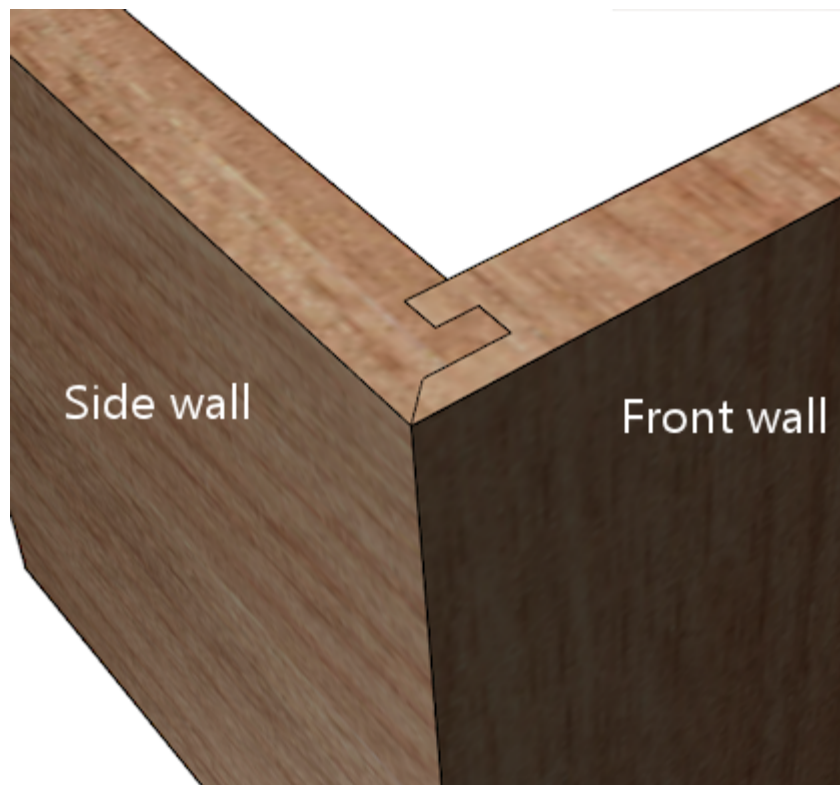
In wood-working, joinery is the art of forming joints where pieces of wood meet.

There's a lot more to this than just nailing boards together; joins can involve angled edges to hide seams, and interlocking tabs and slots to add strength. Joinery is a huge subject that goes well beyond my expertise, so I won't try to offer a primer here. However, I do want to provide a quick overview of how the corner joints are built in the real pinball machines, because you might want to adapt these - either to something simpler or to something better.

Apart from the corner joins, most of the joins we use in the plans are straightforward enough that you probably won't need to change them. Most of the joins (save the corners) are simple dado or rabbet joins that you can execute with straight router bits or a table saw.

The only place in a pin cab where fancy joins are called for is at the corners of the main cabinet. The front corners in particular are prominently visible, so you'll want them to look nice, and they need to be fairly strong, given how heavy a pin cab is. There are several good ways to do these joins, and even the commercial manufacturers haven't settled on a single best way - they've used a number of approaches over the years. I'll go over the details for several good options below.

Locking miter: This is the corner join used on the WPC cabinets of the 1990s. It's called a "locking miter" because the outside edges meet at a 45° angle (that's the "miter"), and it has a sort of jigsaw-puzzle pattern of interlocking tabs and slots that align the pieces and hold them together (the "lock").

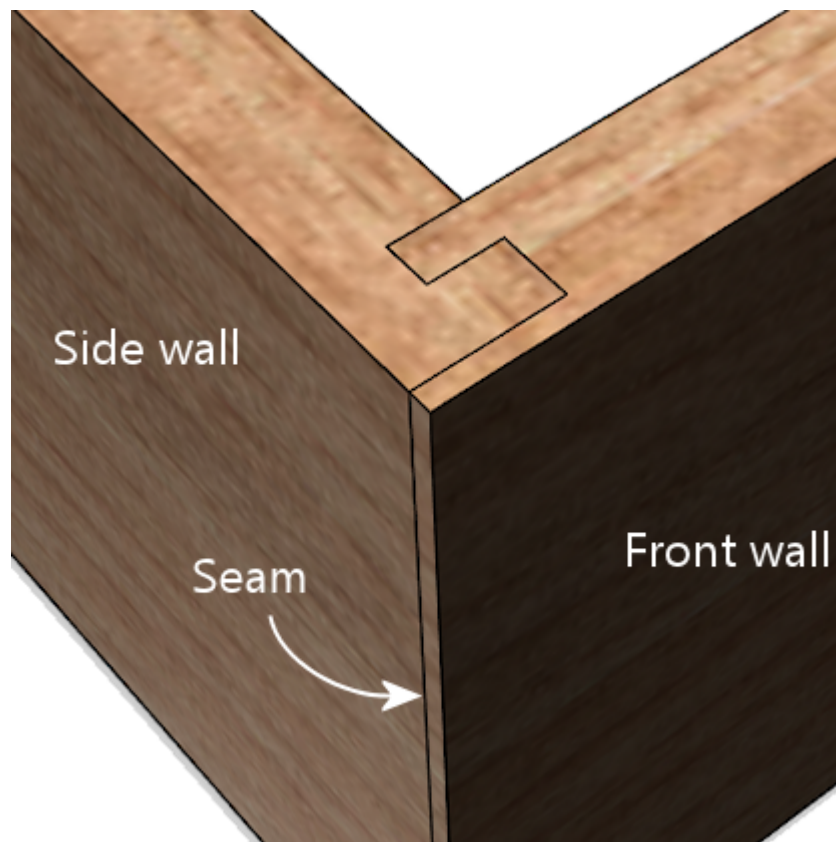


Locking miter join, shown at the front left corner of the main cabinet. This is the type of join that Williams used for the original WPC cabinets from the 1990s.

This is a really nice way to make your corners. The mitered corner makes the seam invisible, and the join is very strong when glued thanks to all of the surface area in the interlocking tabs. You can see from the diagram that that shape is a little complicated to cut, but it's surprisingly approachable, even if you don't have a lot of woodworking experience. For a complete recipe, using a table saw and router table, see Appendix 12, Lock Miter I: The Plywood-Friendly Way. There's also an alternative approach that uses a special-purpose router bit, explained in Appendix 13, Lock Miter

II: The Special Router Bit Way. The first approach works a lot better with plywood, so I think it's the right one for a pin cab project.

Locking rabbet: This is essentially a simplified version of the locking miter joint that dispenses with the 45° bevel at the corner.

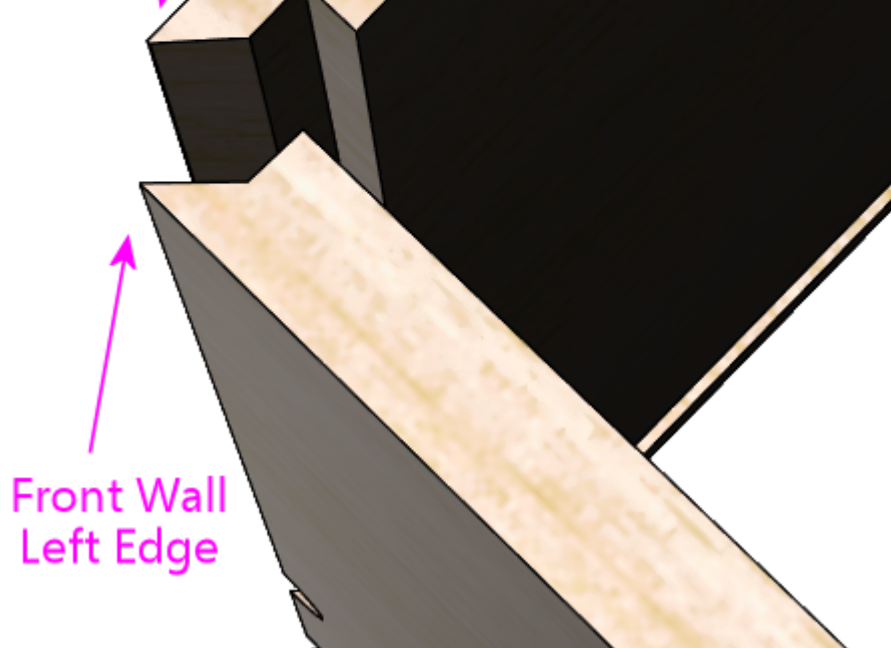


This joint was used on many commercial machines of the 1980s, including many Williams System 11 machines. It has the same self-aligning and self-squaring advantages as the locking miter joint. It's a step down aesthetically, since there's a seam along one side, but that can be minimized by making the front tab fairly thin. The trade-off is that a thinner tab is more delicate prior to assembly, so you have to be careful handling the piece. This joint is quite a lot easier to execute than the locking miter; it only requires three cuts at each corner.

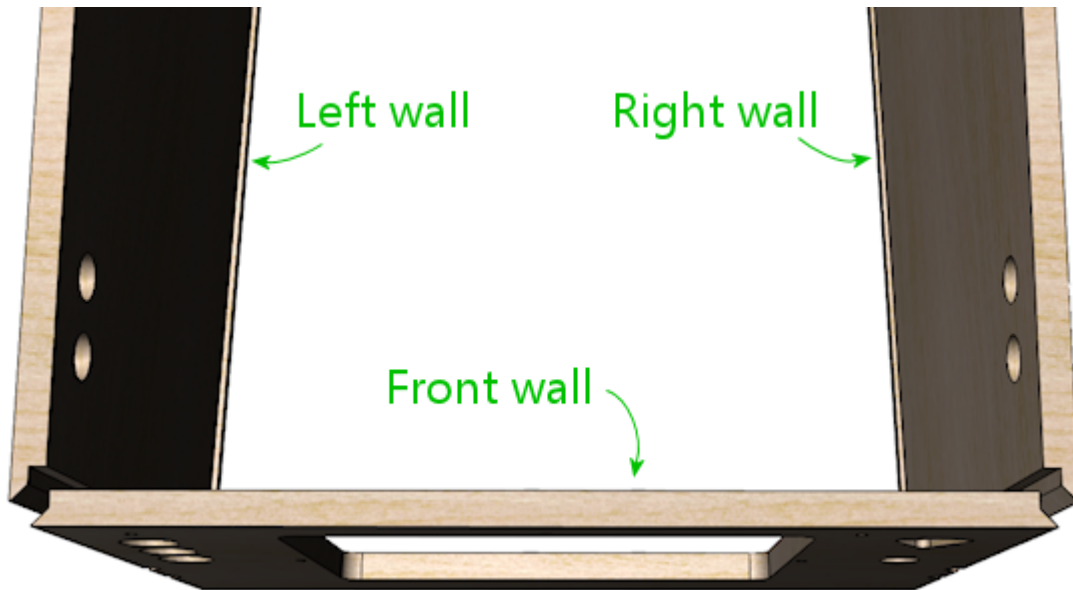
I haven't written a recipe for this joint, but it's easy to find Web tutorials, since it's used a lot in mainstream cabinetry, especially for drawers. Search for "locking rabbet joint" or "locking drawer joint".

Mitered rabbet: This joint has a mitered corner like the lock miter, but it dispenses with the interlocking tabs, and uses a simpler "rabbet" pattern instead.





Mitered rabbet join, at the corner between the front wall and left wall of the cabinet.



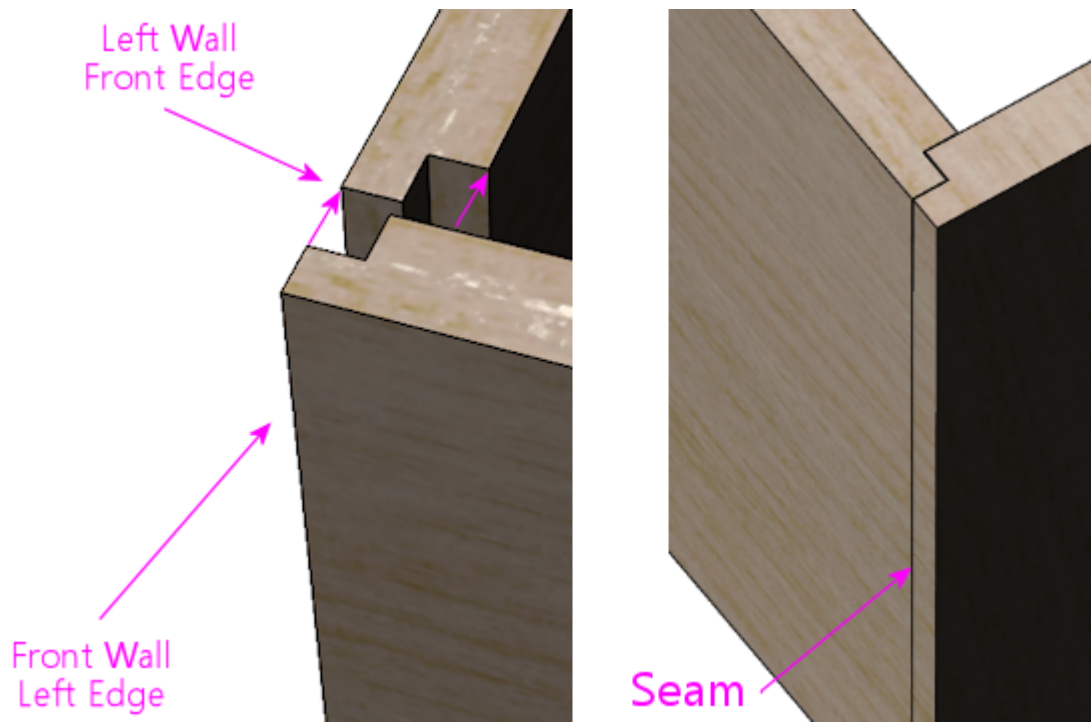
Top view of the front section, with a mitered rabbet at each corner.

The mitered rabbet has the same aesthetic advantage as the lock miter, in that it places the seam exactly at the corner. It's also fairly strong when glued.

You can make a mitered rabbet using either a special router bit set or just a table saw. I haven't attempted either myself, so I won't try to provide instructions, but you can find tutorials on the Web. Search for "mitered rabbet with table saw" or "mitered rabbet router bit". The difficulty level seems similar to that of the lock miter, but it doesn't require as many separate steps, so it's at least a little less labor-intensive.

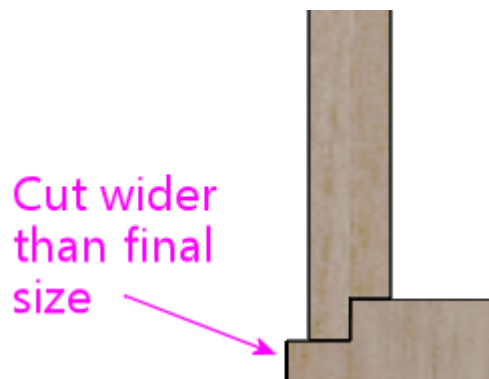
Double rabbet: This is a simpler option that you can make with a table saw or a straight router bit. The double rabbet join dispenses with the diagonal cut out to the corner, and instead uses square interlocking notches. It's easier to construct, but it has a couple of drawbacks. For one, it leaves a visible seam along one of the joined faces. For another, it makes it a little trickier to translate the cabinet measurements to the wood pieces, because of the way one piece slightly extends the apparent

length of the adjoining piece.



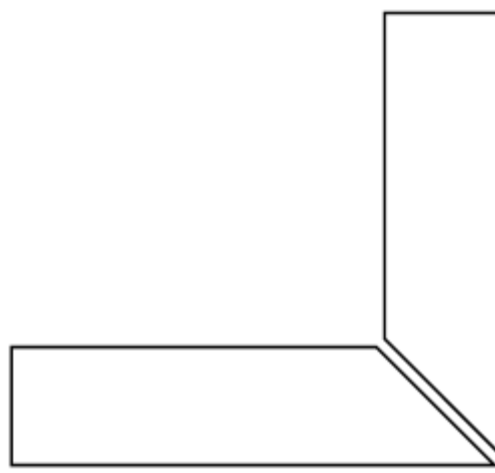
Double rabbet join, similar to the join used in Williams System 11 cabinets. This is a simpler alternative to the lock miter join used in the WPC cabinets, but it has the drawback that it leaves a seam along one face near the corner.

If you decide to use the double rabbet join, there are a couple of things you can do to minimize the visibility of the seam. First, choose the placement of the seam so that it's on the less visible face. The seam only affects one or the other adjoining face at each corner, so you have a choice of which wall will have the seam. The Williams System 11 cabinets placed the seams on the sides (rather than the front face), which seems like the better choice aesthetically, since the front is more visible. Second, cut the front piece a tiny bit wider (1/16", perhaps) than the final size, so that it leaves a little overhang when initially assembled, as illustrated below.



After assembly, the overhang lets you sand down the excess material until it's exactly flush with the adjoining section. It's almost impossible to get the surfaces perfectly flush in the initial cut, so your best bet is to start with a slight overhang that you can sand until flush. You can then add wood filler at the seam to further smooth it out.

Simple 45° miter: Some pin cab builders simply cut the ends of the main cabinet walls at a 45° bevel angle, for plain miter joins:



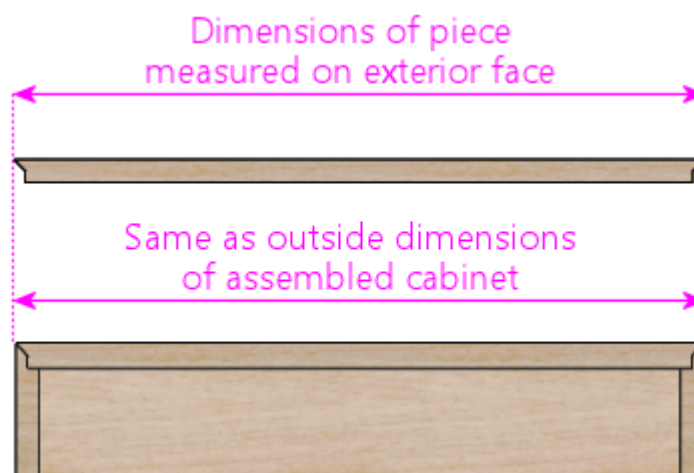
Woodworkers generally consider this an inferior join for large cabinets, since glue is weak when joining end-grain to end-grain like this, and because it's difficult to get the pieces aligned and squared properly given the lack of any interlocking structure. Even so, it might be viable for a pin cab if you're using the new-style Williams leg brackets (part 01-11400-1), since they add a lot of corner strength. I wouldn't personally use this join, but it's an option if you want to simplify the woodworking.

Adjusting dimensions for joinery

Pay close attention to the effects of your chosen corner joins on the overall dimensions.

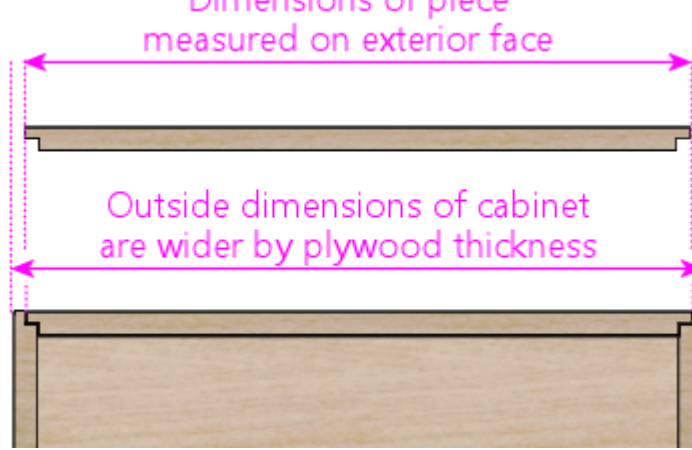
The dimensions shown in our plans assume that you're using a mitered join of some kind for the main cabinet corners. Our illustrations show those corners with a mitered rabbet, so you'll see that join in the close-ups. Any miter join, including the locking miters and the simple 45° miter, is equivalent in terms of all of the measurements, so there's no need to make any adjustments for those.

With the mitered joins, note how each individual piece's dimensions exactly match the assembled cabinet's outside dimensions for that section:

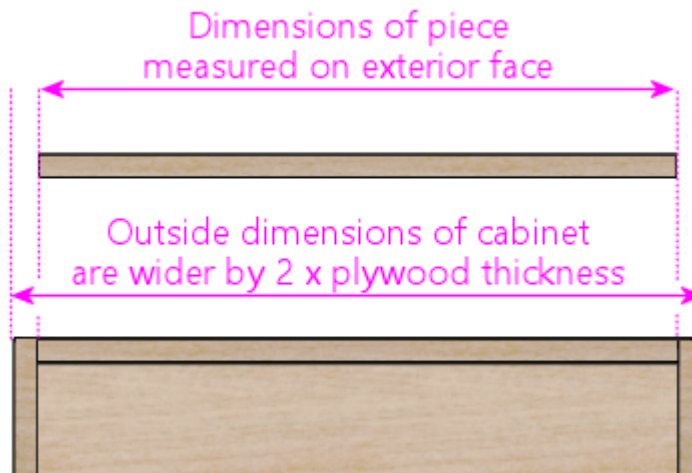


In contrast, with the rabbet join, note how the "inside" piece (the one forming the face with the seam) is slightly shorter than the assembled cabinet's outside dimensions. This will be shorter at each corner by the depth of the rabbet groove, which is typically half the plywood thickness, so assuming there's a join like this at each end, the overall piece will need to be cut shorter than the desired final outside dimensions by $2 \times \frac{1}{2} \times \text{the plywood thickness} = 1 \times \text{the plywood thickness}$:

Dimensions of piece



Likewise, for a butt joint, the inside piece will need to be shortened by $2 \times$ the plywood thickness, compared to the finished outside dimensions:

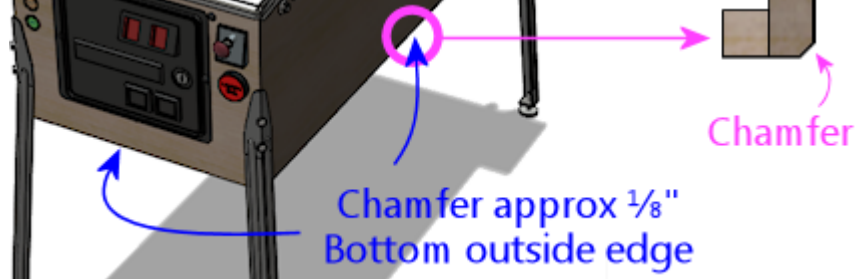


Our measurements for the main cabinet are based on using mitered joints at the visible corners, so be sure to adjust the dimensions before cutting if you're using a different joint.

Edge finishes

On the original WPC cabinets, the outside bottom edges of the side and front walls are finished with a chamfer (a 45° bevel), about $\frac{1}{8}$ " wide. I don't think they did that for looks, but rather to soften the plywood edge, to make it less sharp and splintery. On my cab, I kept it simpler and just sanded the edges smooth. If you do decide to apply a chamfer with a router bit, it might be a good idea to test the bit on a piece of scrap plywood first - I've read that some plywood will chip if you try to bevel the edge like this, and that would defeat the whole purpose of smoothing it.

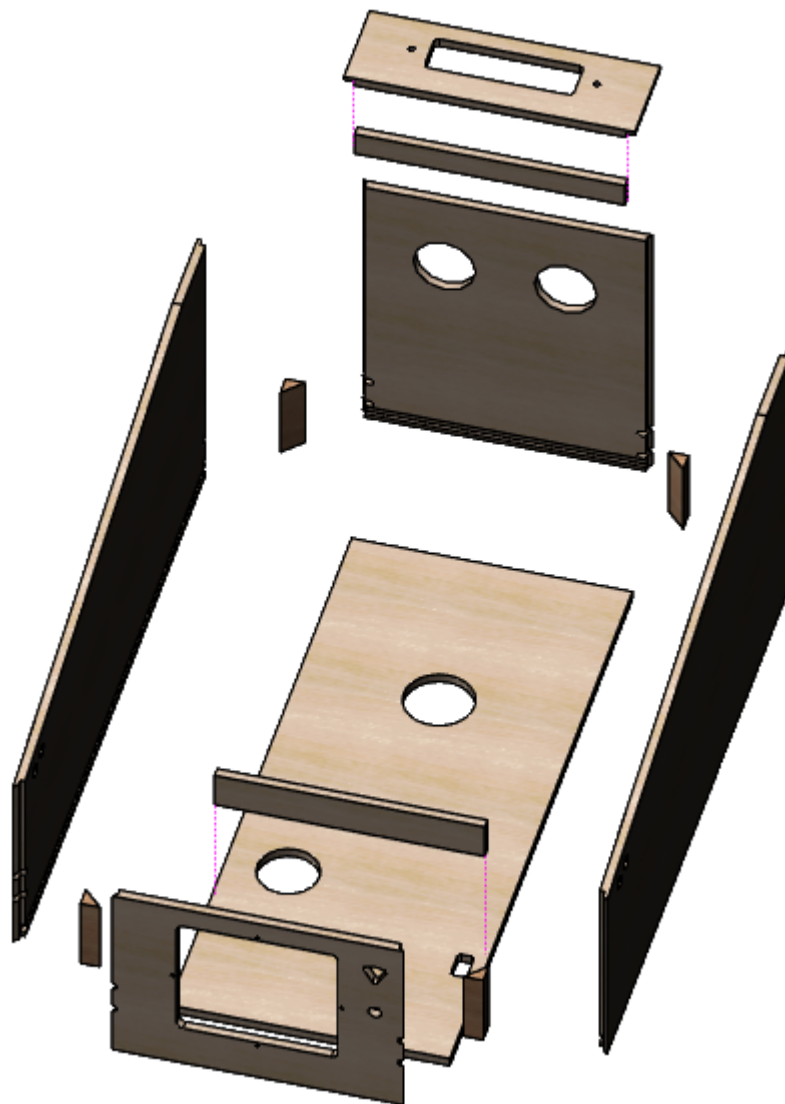




On the WPC machines, the front vertical edges (at the corners between the front wall and the left and right walls) are square, without any rounding or beveling. Some of the newer Stern machines round those edges out slightly - it looks like they route the edge with a 1/8" roundover bit. I just lightly sanded the corners on my cab until they felt smooth.

Exploded view

This view shows all of the pieces making up the main cabinet body.



The triangular wood pieces at the corners go under the metal brackets the leg bolts screw into. They provide reinforcement at the corners (to prevent the corners from splitting) and help strengthen the leg attachment. The leg bolts (two per corner) go through these at a 45° angle.

The two pieces at the top rear form a "shelf" that the backbox rests on. The

rectangular routed opening in the horizontal piece is to pass power and video cables between the cabinet and backbox. The opening shown is what's used on the real machines, and it works well for a virtual cab as long as you only need to pass cables through. You might need a larger opening, though, if you plan to use a large monitor in your backbox that needs to extend into the main cabinet. This isn't an issue for a typical three-monitor setup with a laptop display for the DMD (or a real DMD device).

The smallish slat near the bottom front attaches to the floor on the real machines to form a niche to hold the cashbox. (The cashbox sits under the coin slots to collect the inserted coins.) Most virtual builds omit this piece to leave more room for the PC motherboard, which most people situate on the floor of the cab about halfway back.

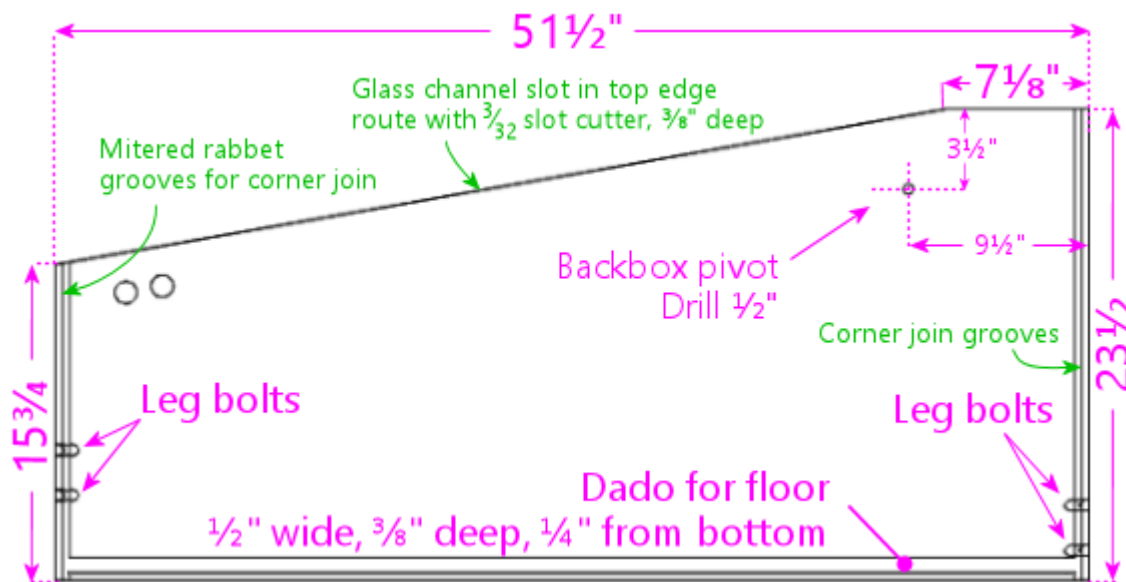
Cutting up the plywood

There are numerous ways to divide plywood sheets into the panels making up the cabinet. For some suggested layouts, see Appendix 9, Plywood Cutting Plans for Cabinet Construction.

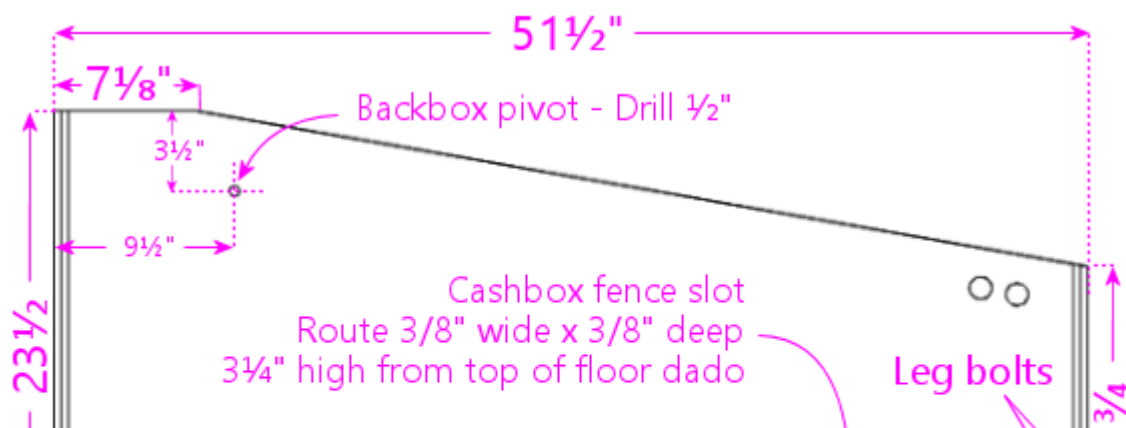
Side walls

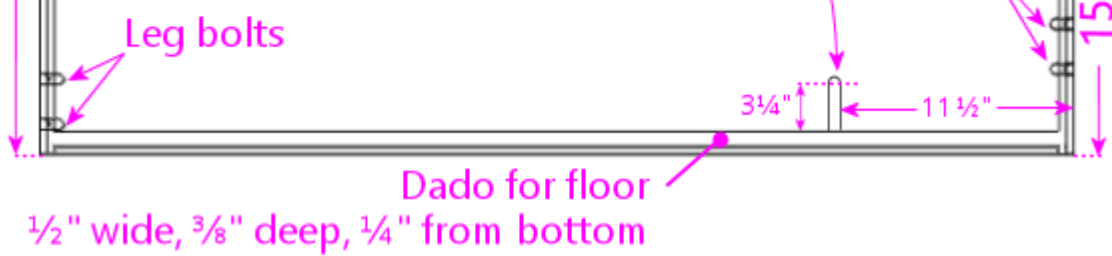
Here are the side walls. The views are from the interior of the cabinet, to show details on the joinery routing.

(The flipper button holes and leg bolt holes are marked, but for the sake of readability, the dimensions aren't shown here. We'll provide close-up diagrams for these elements, with all of the measurement details, later in the section.)



Left side wall, viewed from the cabinet interior side





Right side wall, viewed from the cabinet interior. The right wall is a simple mirror image of the left wall.

Remember that we're measuring the dimensions of the pieces based on a mitered join (either a mitered rabbet or lock miter) at the front and rear corners, meaning that the piece's dimensions match the outside dimensions of the assembled cabinet. If you're using a different join at the corners, be sure to make any necessary adjustments. See Joinery above.

Some more views to help with visualization:

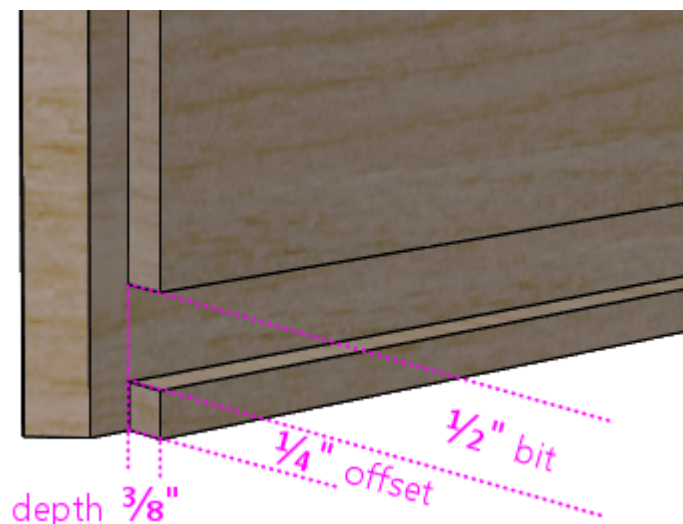




Backbox hinge pivot: The backbox pivot is a $\frac{1}{2}$ "-diameter drilled hole for attaching the WPC-style backbox hinge. If you're using a different hinge system to attach the backbox, omit this.

Note: Some people prefer to wait to drill for the hinge pivots until after assembling the cabinet and attaching the hinges to the backbox, so that they can drill the pivots based on the actual assembled alignment of the backbox. The procedure I've always used is basically the opposite: drill the hinge pivot first, attach the hinges there, and then drill the backbox bolts for the hinges based on the final alignment. I haven't tried it the other way, so I'm not sure if that would be easier or harder overall, but the basic idea is the same either way. Use your discretion as to which approach sounds better.

Floor dado: The dado at the bottom is for the cabinet floor. Use a $\frac{1}{2}$ " straight router bit to cut a groove $\frac{3}{8}$ " deep (halfway into the thickness of the plywood), parallel to the bottom edge of the wall, $\frac{1}{4}$ " from the bottom edge. This is on the **inside** face of the wall; the edge of the cabinet floor fits into this groove when assembled.



*Left cabinet wall showing the dado (groove) for joining with the cabinet floor.
Route the dado with a $\frac{1}{2}$ " straight bit to $\frac{3}{8}$ " depth, $\frac{1}{4}$ " (or $\frac{3}{8}$ ", if you prefer) from*

the bottom edge. This groove runs the whole length of the side wall. This is on the interior face, since it joins with the cabinet floor. The diagonal/step shape along the vertical edge at the left is the mitered rabbet cut for joining to the front wall, illustrated in more detail above.

Note: Some other people's WPC-replica plans show the floor dado at 3/8" from the bottom, rather 1/4" as depicted above. My original WPC cabinets measure 1/4", but I measured one older System 11 machine at 3/8". The larger 3/8" offset will make the joint a little stronger, and shouldn't much affect anything else, so I don't see any downside; use your discretion. Whatever you decide, be sure to route the corresponding dados in the front and back walls at the same offset, since they all have to align when assembled.

Cashbox fence slot: The slot for the cashbox fence is only needed if you plan on installing said fence, which is useful if you're going to use the standard type of coin collector box ("cashbox") made for commercial pinball machines. The cashbox sits at the front of the cabinet under the coin slots, and the fence helps hold it in place.

Most virtual cab builders don't use the standard cashbox because it takes up so much space. I'd omit the fence if you're not going to use the standard cashbox.

The routed slot isn't strictly necessary even if you do include the fence, but it makes it easier to install the fence during cabinet assembly by providing an anchor point to glue it to.

The slot only goes in the right wall. You can move it to the left wall if that's more convenient - it really doesn't matter which side it's on. But you only need a slot on one side or the other.

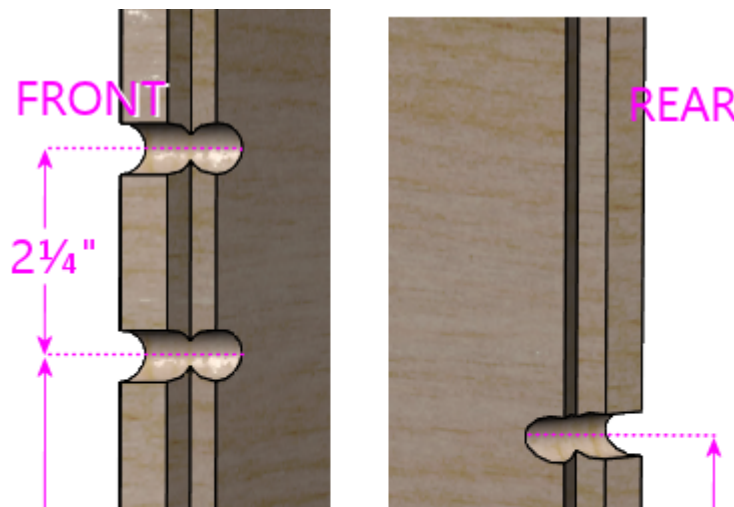
If you're going to use a custom cashbox that's not the standard size, you should move the fence (and thus the fence slot) to match the depth of the box.

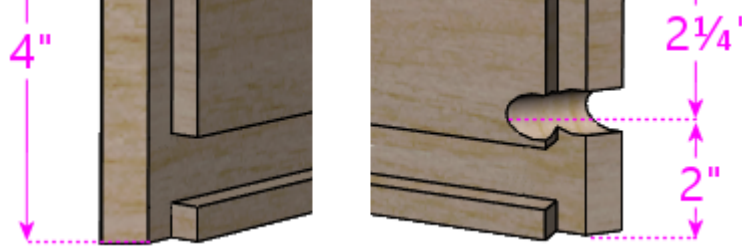
Edge finishes

The original WPC cabinets use a slight chamfer (a 45° bevel) on the outside bottom edges of the side walls, to soften the edge and reduce splintering. This is optional, but it will reduce the chances of snagged clothes and cuts from bumping into the side. See Edge finishes above.

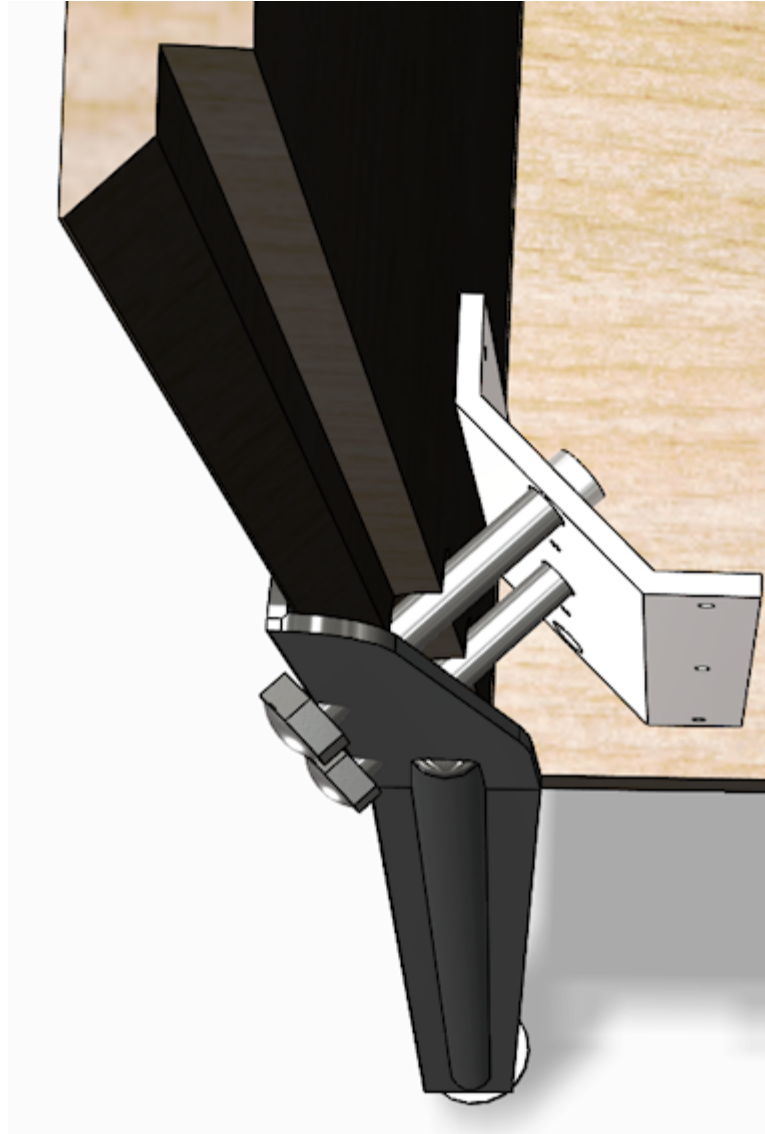
Leg bolts

The leg bolt holes are a little tricky. The bolts go through the corners at a 45° angle, so they bore through both adjoining walls at each corner. So, as shown in the illustration, the left and right walls only have "half a hole" for each bolt - really more of a semicircular notch.





Leg bolt holes, front (above left) and rear (above right). The distances are shown from the bottom of the cabinet. Note that the front legs are mounted higher on the wall than the rear legs. The legs themselves are the identical parts front and back, so the different mounting position is used to give the cabinet its characteristic tilt angle. The bolts are 3/8" diameter.



Cutaway view (with the front wall removed) showing the leg bolts installed, to better illustrate how the bolt holes intersect the side wall. The triangular wood piece that normally fills the gap between the metal plate and the inside wall is also hidden.



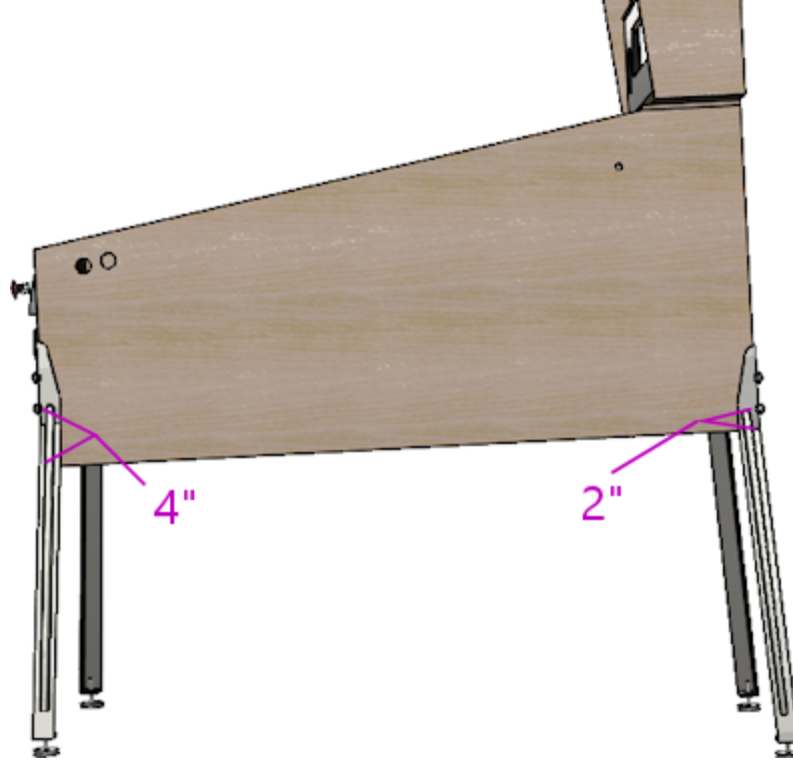


Illustration of how the leg install positions affect the cabinet slope. The legs are mounted higher on the cab in the front, which effectively raises up the back end slightly to slope the machine down toward the front. Standard pinball legs come with adjustable foot pads that you can use to make sure all four legs touch down and to fine-tune the playfield slope. The slope isn't needed for "physics" reasons on a virtual machine, but it's still desirable for an authentic appearance, and it also improves the viewing angle for the main TV.

There are two approaches to drilling the holes:

- **Before** assembly, by cutting half-cylinder notches in each panel. The bolts are 3/8" diameter, so each notch needs to be 3/16" deep and 3/8" wide. You could accomplish this by hand with a round file, or using a router with a 3/8" round-nosed router bit. A round-nosed bit has a half-dome shape for its tip, so it routes half-cylinder grooves just as we need; route to a depth of 3/16".
- **After** assembly, using a regular drill with a 3/8" bit to drill the holes. It's difficult to drill into a corner at a 45° angle free-hand, but it's doable using a "drilling block" - a tool with guide holes for different size drill bits. You'll need one that's designed for drilling into a corner; you can find these on Amazon by searching for "corner drill guide" or "45 degree drill guide". Some people have built DIY guides by bolting together a couple of 2x4 pieces to form a corner.

If you prefer something more purpose-built, here's a 3D-printable model for a drill guide that's designed just for the leg bolts. It fits over the corner and has two 3/8" holes at the standard 2¼" spacing.

leg-bolt-drilling-jig.zip





Preparing to drill for a leg bolt at the front right corner, using a general-purpose drill guide block. This drill block features a notch specifically for drilling into a corner at a 45° angle. I'm using a band clamp wrapped around the whole cabinet to hold the drill block in place. You have to clamp the drill block down pretty tightly, and even then you have to be careful to use a steady hand - those 45° notches are small, and the drill gives you a lot of leverage.

Personally, I find the "before" approach too difficult to do by hand, because of the 45° angle and because you have to get the notches on the adjoining edges align perfectly. This is probably only workable if you're making the panels with a CNC machine. I'd go with the drill-after-assembly approach otherwise.

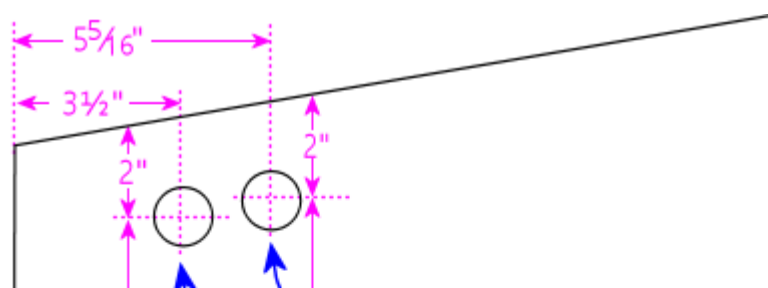
With either method, the holes should end up being a tight fit for the bolts. That's good, since you don't want the legs to be wobbly on a 250-pound cabinet. But if they're too tight, try rubbing a little paraffin wax or a similar dry lubricant on the bolts. (I wouldn't use anything oily or greasy.) If that still doesn't work, you can use a small round file to expand the holes slightly - but as little as possible, to avoid weakening the corner.

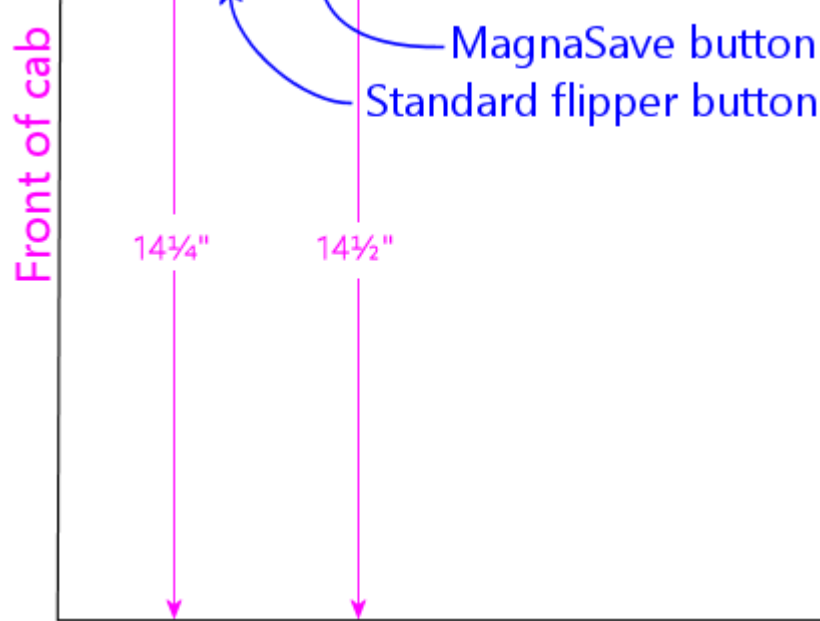
Flipper buttons

Here's the drilling plan for a set of two flipper button holes on each side. The front button in each set is the regular flipper button, and the rear button is the "MagnaSave" button, which is for the benefit of some pinball games that have extra controls beyond the regular flipper buttons. The rear buttons are optional, and not everyone likes them since they're not all that common on real machines, but I think it's good to include them because of the large number of virtual tables that make use of them. See Appendix 4, Tables with MagnaSave Buttons for more about these buttons, and a list of some of the tables that use them.



Note! Some side rails are wide enough to cover the flipper buttons, in which case they'll come with pre-drilled holes for the buttons. The WPC rails are narrow enough that they sit entirely above the flipper buttons, so they don't need any holes for the buttons. If you're using wide rails that do cover the flipper button area, ignore our drilling locations! Your cabinet flipper button holes need to line up with the ones in the rails. So use the button holes in your rails to determine where to drill.





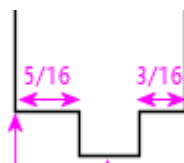
Flipper button drill hole detail for WPC-type side rails. Measurements are in inches; distances are to the center points of the holes. (Don't use these locations if you're using older side rails that cover the flipper buttons. Instead, use the pre-drilled flipper button holes in your rails as drilling templates, so that the cabinet holes line up with the holes in the rails.)

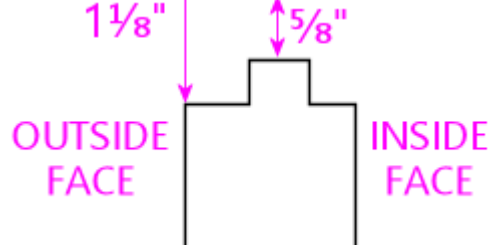
The distances in the diagram are measured from the center of the drill holes to the front and top edges of the wall, square with the front edge. The measurements are referenced to the **outside face** of the **fully assembled** cabinet. If you're measuring prior to assembly, make any adjustments needed to account for offsets from your front corner joins. Mitered joins shouldn't require any adjustments, since the outside edges of all faces go all the way to the corners.

Don't rely on the locations in the diagram if you're using wide side rails that extend over the flipper buttons. Those come with pre-drilled holes for the flipper button, so you'll need the cabinet wall drill locations to match the pre-drilled rail holes. Do a dry fit with the rails to determine the drilling location.

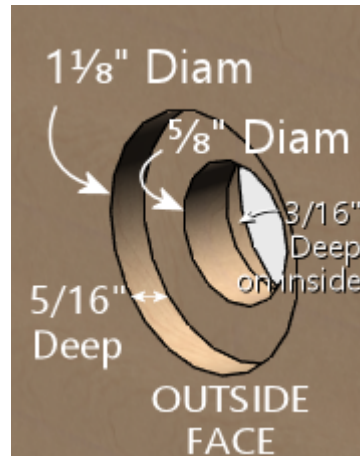
How to drill:

- The easy way: drill straight through with a 1 ¹/₈" diameter hole saw or Forstner bit. This works only if you're using something to anchor the button on the inside, such as the VirtuaPin flipper switch bracket or an LED board (see Chapter 55, Button Lamps) to illuminate the button.
- The original way (used on most of the real machines): This pattern has a narrow waist for the stem of the button, and larger insets on the outside and inside for the body of the button and the Pal nut, respectively.
 - Drill a small pilot hole (1/8") on the center, all the way through
 - Use a 1 ¹/₈" hole saw, Forstner bit, or router bit to drill a 5/16"-deep depression from the **outside**, on the same center
 - Use the same 1 ¹/₈" bit to drill a 3/16"-deep depression from the **inside**
 - Drill the rest of the way through with a 5/8" bit, on the same center





Schematic diagram of the "original" flipper button drilling pattern. This is an edge-on view of the side wall.



The original "stepped" pattern lets you fasten the button with a Pal nut, without any additional brackets on the inside. Use this pattern if you don't plan to use an LED board or switch holder bracket. The straight-through approach is better if you're planning to use an LED board to illuminate the button, since it provides a tunnel for the light to shine through. But you need some sort of bracket on the inside in this case, because the Pal nut fits through the larger $1\frac{1}{8}$ " hole. An LED board can serve as the bracket, as will a VirtuaPin flipper switch holder. If you're not planning to use one of those, the original stepped pattern is better.

Variations:

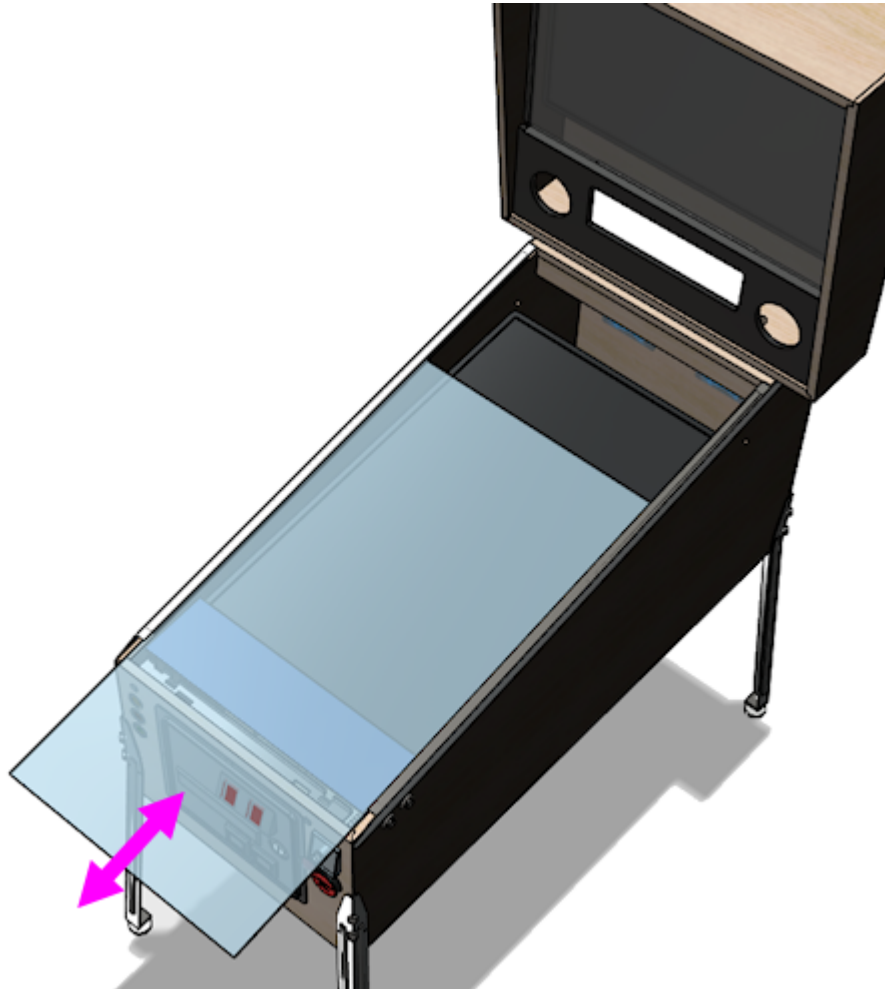
- The rear (MagnaSave) buttons are optional. If you don't want to include them, simply don't drill the holes. The regular flipper buttons go at the same position whether or not you include the MagnaSave buttons.
- There are at least two other good ways to position the MagnaSave buttons. Some people place them directly below the flipper buttons, and some people prefer them diagonally behind and below the flipper buttons. Both of those patterns have precedents in real pinball machines that had the extra buttons (see Appendix 4, Tables with MagnaSave Buttons). The layout in my diagrams is based on the Williams MagnaSave games from the 1980s, so it's probably the most familiar look to most players, but not everyone likes the feel, due to the stretch to reach the rear buttons. The more vertical layouts are arguably easier to reach, and make it easier to keep a finger on each button.
- Williams System 11 games (1980s) placed the flipper buttons about $\frac{1}{4}$ " higher than shown in my diagrams, which are based on the WPC games (1990s). System 11 games used broader side rails that covered the flipper buttons, so I think the slightly different positioning is purely to accommodate the different rails. I don't think it noticeably affects the feel.

Glass channel slots

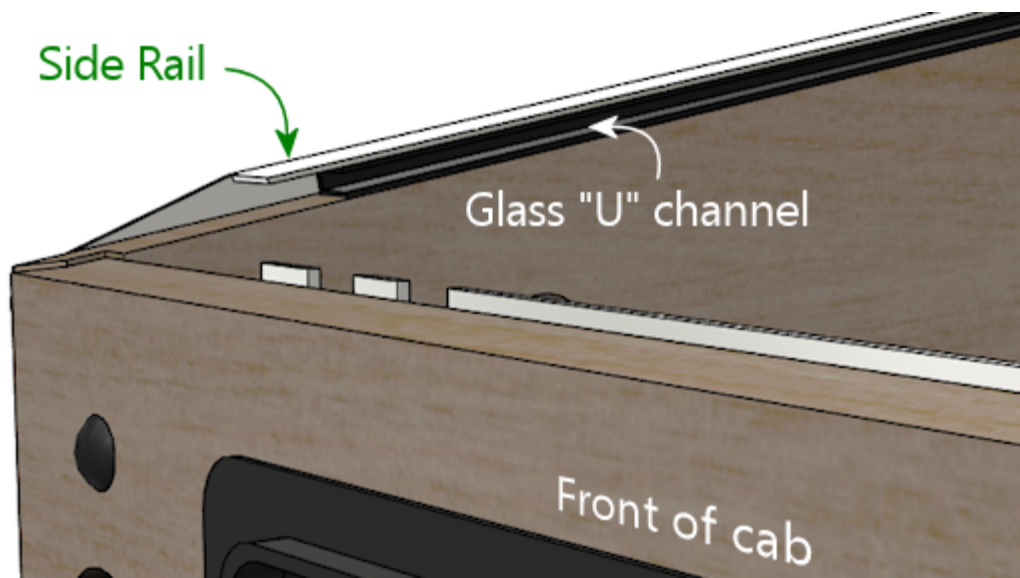
If you're going to install the standard side rails and a glass cover over the playfield,

you should also install a set of "glass channels". These are plastic "U"-shaped trim pieces that fit under the side rails, along the left and right edges. These hold the glass at the sides.

Because the glass channels are "U" slots along the length of the machine, you can slide the glass in and out of the channels through the front of the machine, after removing the lockbar. This is part of the tried-and-true design of the real machines that lets an operator easily open up the machine for maintenance access, and I think it's a great thing to replicate in a virtual cab.

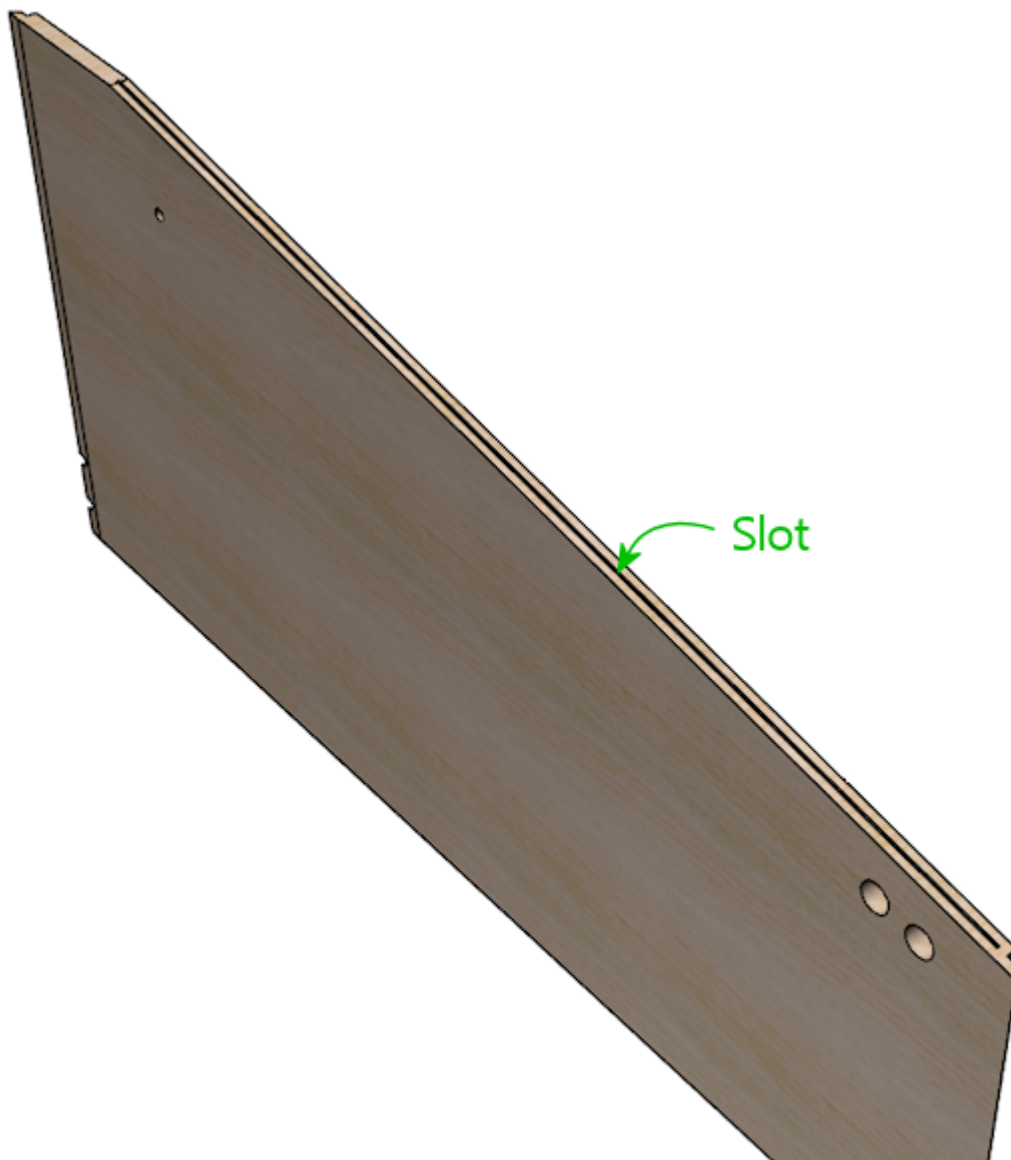
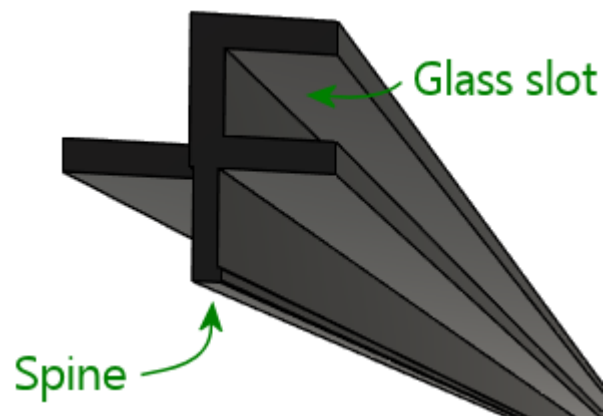


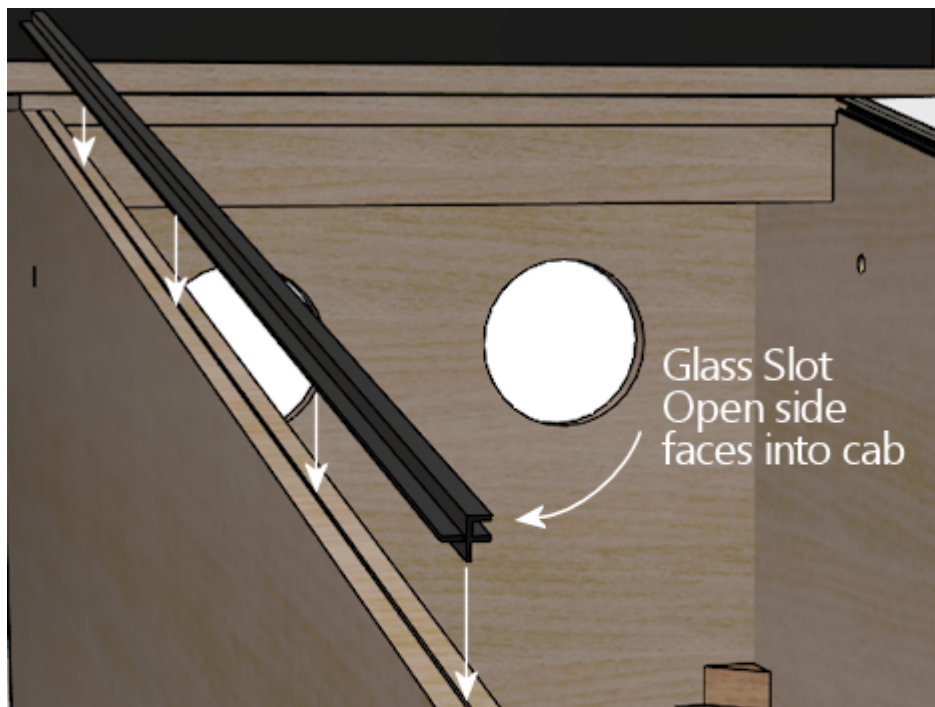
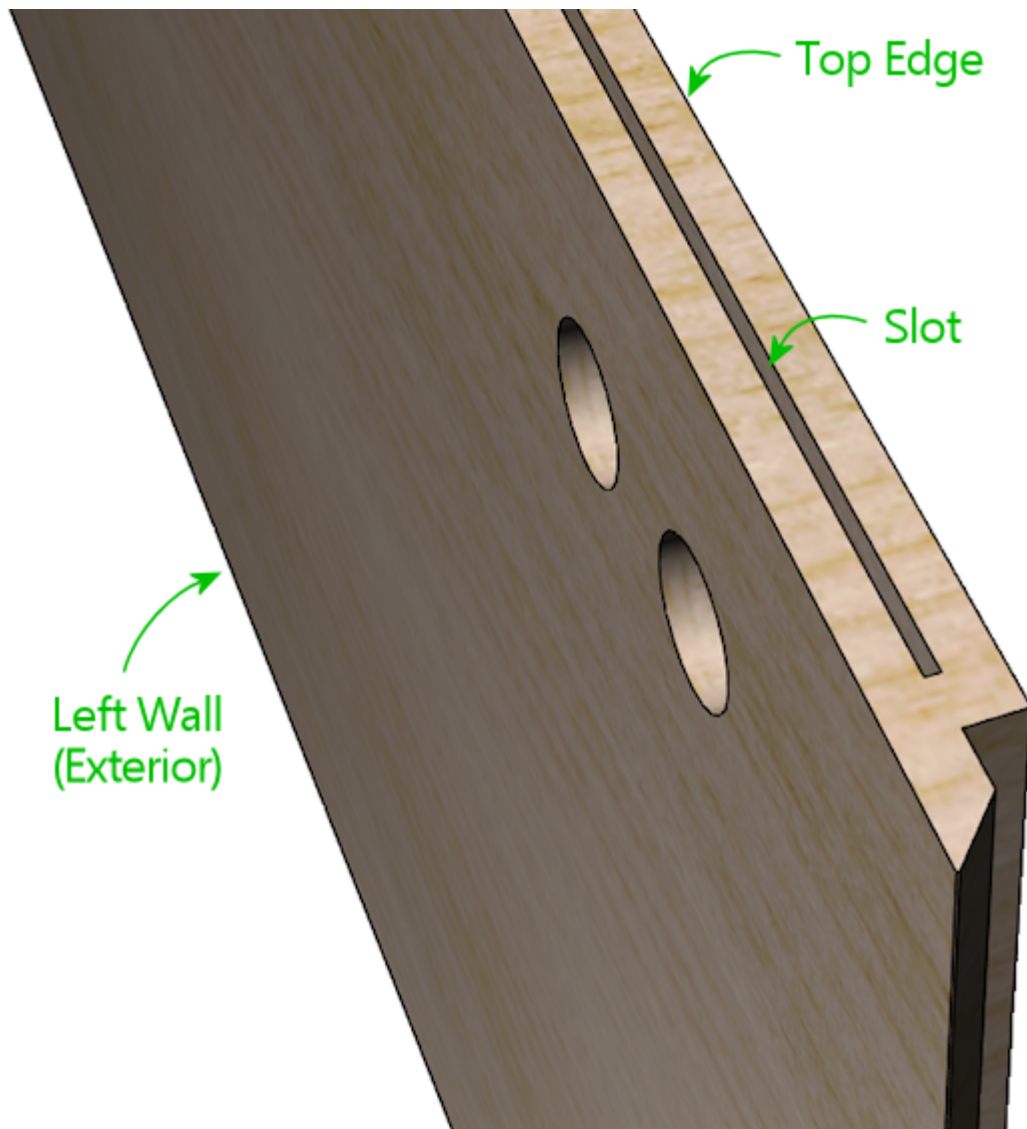
The glass channels are installed under the side rail. Here's a close-up of how they look when installed:

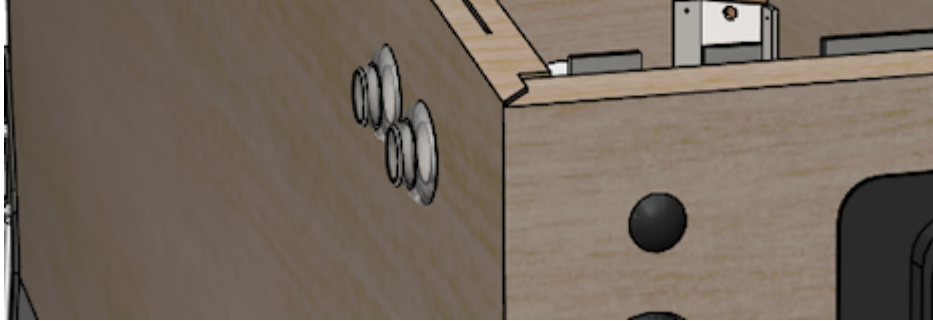




The channels attach to the side wall via a "spine" sticking out of the bottom of the plastic channel. The spine which fits into a slot in the top edge of the side wall.







This is a neat design, in that you don't need any fasteners or adhesives. You just press the spine into the slot, and it's held there by friction. If it's ever necessary to take the channel out, you can just pull it out. On the other hand, it presents us with another little wood-working challenge: how do we cut that precise little slot?

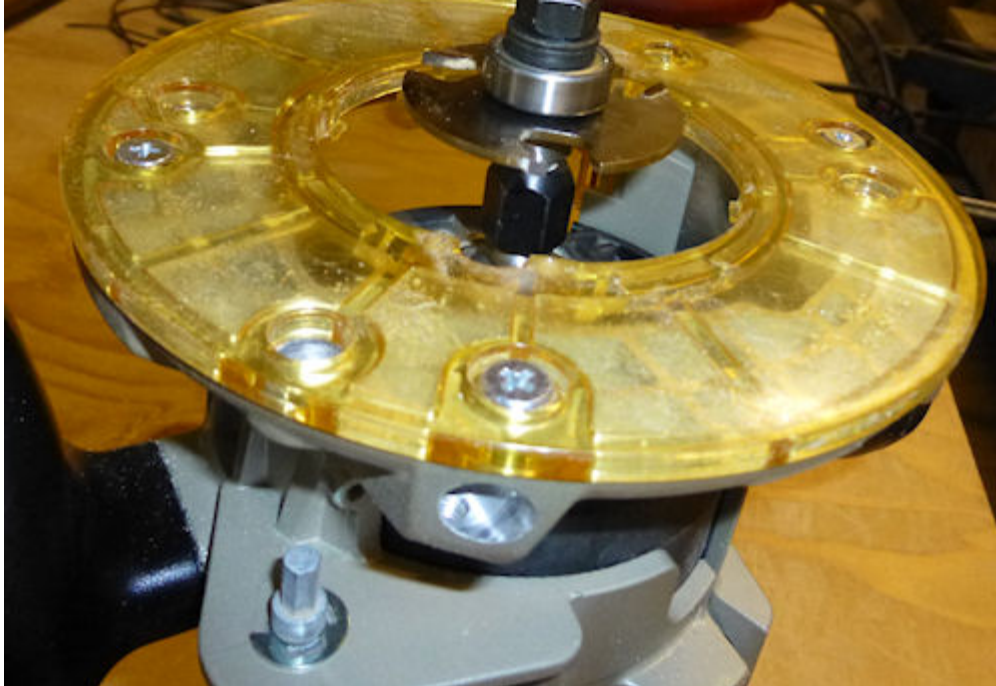
As usual, it turns out that there's a special tool for this job, and it's really easy once you have that magic tool. What you need in this case is a special-purpose router bit called (naturally) a slot cutter. Just as a drill bit is designed to drill a hole of a specific diameter, each slot cutter bit is designed to make a slot of a specific width and depth. For this job, you need a bit with a $\frac{3}{32}$ " slot width and $\frac{3}{8}$ " slot depth. (A deeper slot, like $\frac{1}{2}$ " or $\frac{5}{8}$ ", will also work if you can't find a bit for that exact depth. But the width is important - it should be exactly $\frac{3}{32}$ ".) The bit I use for this is Freude part #63-106, which works perfectly.

Once you have the necessary slot cutter bit, cut a slot along the top edge of the sloped portion of each side wall, centered along the edge, starting about $1\frac{1}{2}$ " from the front and ending at the top of the sloped section. The photos below give an overview of how you set up the bit and cut the slot.



Slot-cutter bit, $\frac{3}{32}$ " slot width, $\frac{3}{8}$ " depth (the photo shows a Freude #63-106, but other brands are available with the same specs)





The slot-cutter bit set up in a hand router. This bit works best with a fixed-base router. A plunge router will also work - you just have to lock the depth. You can also do this with a router table, using your router table's setup for a bit with a pilot point. I'm using a hand router for these illustrations, since I've found that to be an easy way to use this bit, but the process is essentially the same with a table.

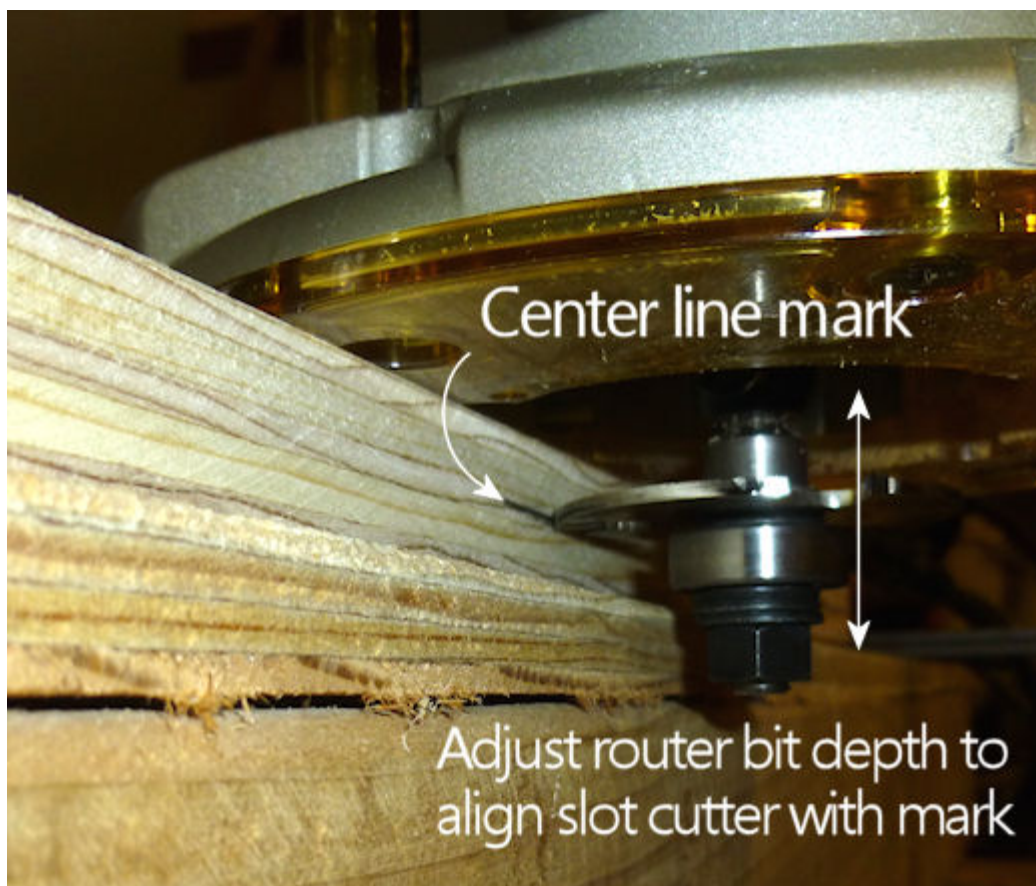


Measure the thickness of the plywood, and mark the centerpoint.





To make sure you found the exact center, flip the board over and measure the same distance from the other side. Adjust your measurement and repeat until the center mark is accurate.



Now clamp the board to a horizontal surface. Make sure the router is unplugged! Place the router base flat on top of the board, with the bit against the edge. Using the router's cut depth adjustment (see your router's instructions), adjust the bit depth so that the slot cutter blade lines up with the center line you marked in the previous step. Lock the router at this depth - this sets the bit to cut the slot at the center of the board's edge.



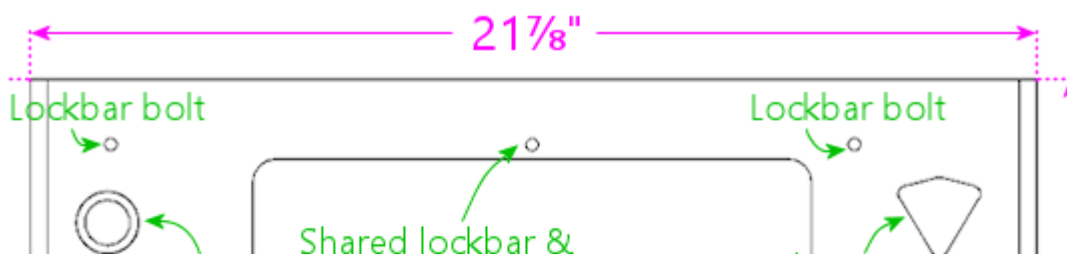


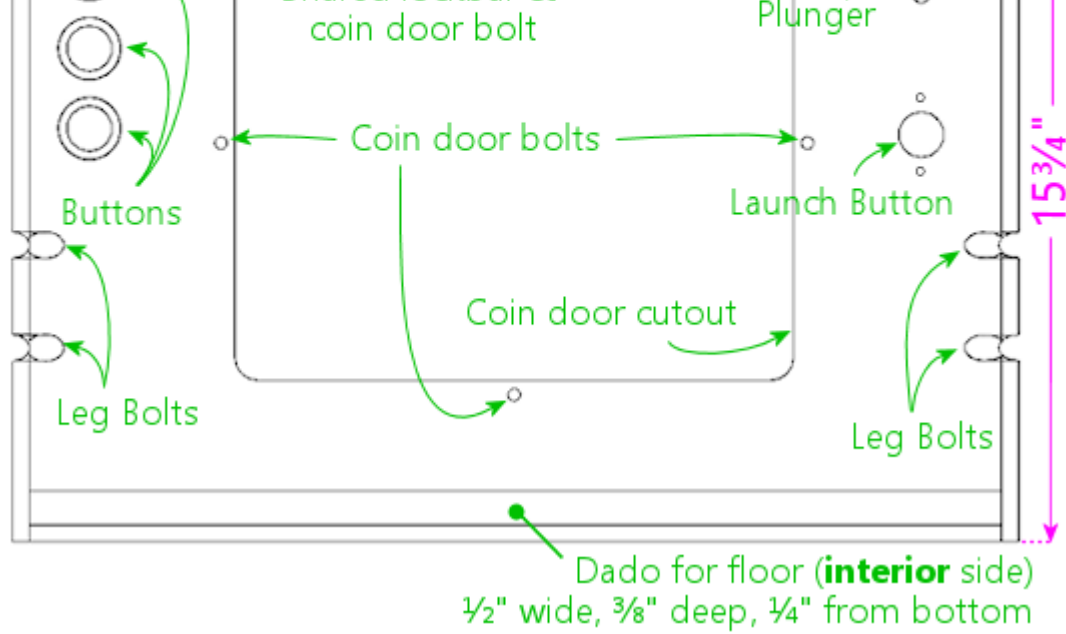
You're all set to cut the slot. Plug the router in. Make sure the board is securely clamped to a horizontal surface. Place the router flat against the board with the bit hanging over the edge right next to the starting point for slot. For safety, always make sure that the bit isn't touching the work piece (or anything else!) when you switch the router on - so position it so that the bit is just clear of the work piece, at the point you want to start the cut, with the base flat against the board. Keep the router base flat against the board at all times throughout this procedure, and hold the router in both hands to keep it steady. When the router is up to speed, gently slide it sideways into the edge of the board to start cutting the slot. The bit's pilot point will automatically stop the bit at the correct slot depth, so just keep sliding it into the edge until it hits the pilot point. Now slowly move the router along the edge of the board, parallel to the edge, keeping the pilot point pressed against the edge, until you reach the end of the span where you want the slot to be. Finally, withdraw the bit from the slot, by sliding the router sideways away from the edge of the board just far enough for the bit to move clear of the slot, then turn off the router.

Front wall

The front wall is the most complex section of the cabinet. It has a whole bunch of things attached: the coin door, several pushbuttons, the plunger, the lockbar, and the leg bolts. There are so many things vying for a limited amount of space that the positioning of each part is pretty constrained; everything fits together like a 3D puzzle.

I initially tried to cram all of the measurements for all of the cutouts into a single diagram, but I quickly abandoned that idea, since it was way too busy. So instead, I've broken it out into several diagrams, one for each set of cutouts. We'll start with the basic outline and its overall dimensions, with the purpose of each cutout labeled.

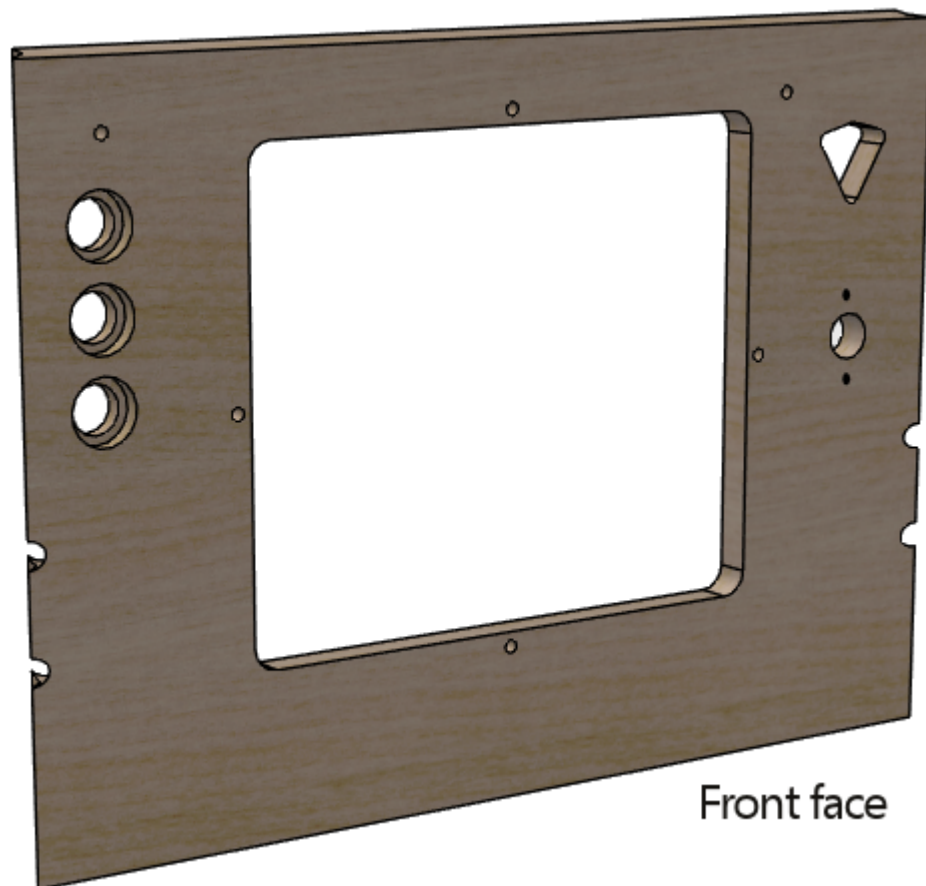


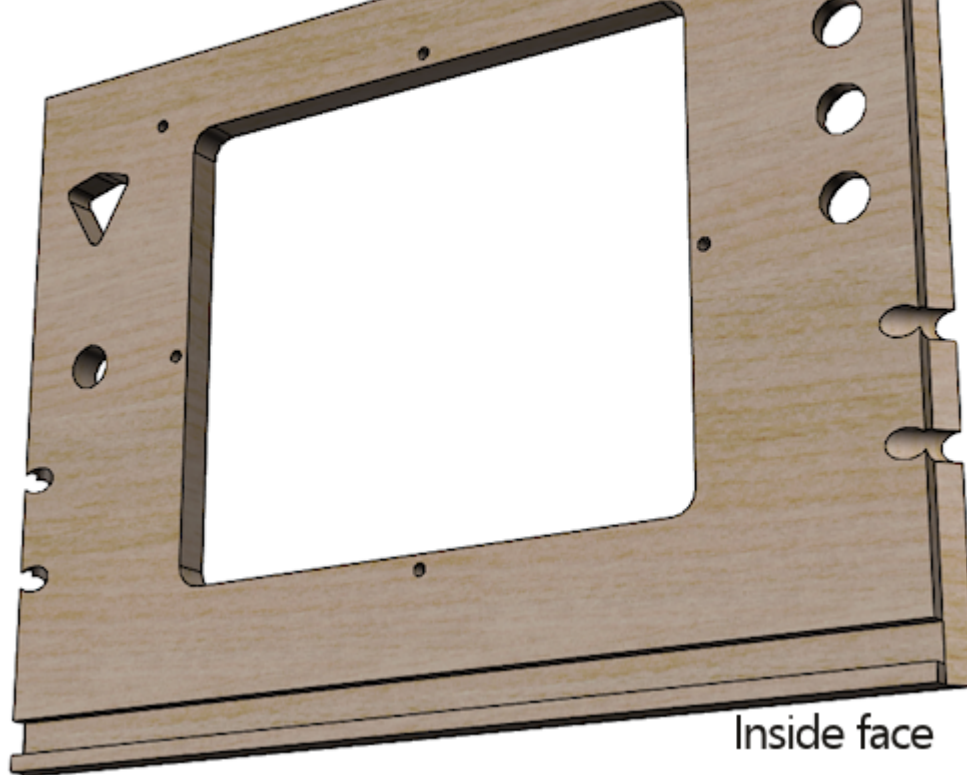


Main cabinet front panel, viewed from the front (exterior side).

Remember that we're measuring the dimensions based on a mitered rabbet join at the corners, and that you might need to adjust the dimensions slightly if you're using a different join style. See Joinery above.

More views for visualization:



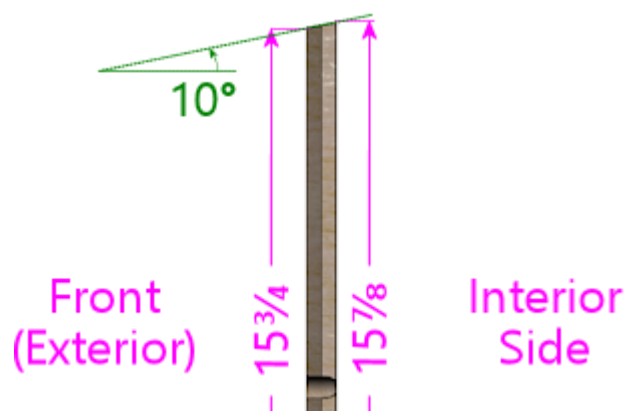


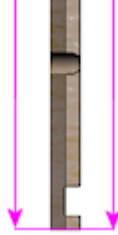
The overall width is based on the standard-body design. If you're building a widebody or custom-width cabinet, adjust the width of this piece accordingly. Keep the coin door cutout centered horizontally at the new width, and keep the buttons and plunger at the same distance from their respective side walls.

The dado at the bottom is for joining with the floor of the cabinet. This is exactly the same as the ones in the side walls: route a $\frac{1}{2}$ " wide groove to a depth of $\frac{3}{8}$ " (half the thickness of the plywood) along the whole bottom edge, on the interior side, $\frac{1}{4}$ " from the bottom. As mentioned earlier, some other published WPC plans offset this dado from the bottom by $\frac{3}{8}$ " instead of $\frac{1}{4}$ "; use the same offset that you used for the side walls.

The leg bolts go through the corners of the front face at a 45° angle, just like the way they work with the side walls. Route notches for the bolts exactly as we described earlier for the side walls. Use the same positioning (measured from the bottom edge) as for the front legs on the side pieces. The front notches in the side walls need to align with the notches in the front wall when the cabinet is assembled.

The top edge of the front wall should be cut at a 10° bevel angle, to match the slope of the side walls. This corresponds to a rise of about $\frac{1}{8}$ over the thickness of the plywood. In other words, the height at the back face (the side facing the interior of the cabinet) should be about $\frac{1}{8}$ taller than the height at the front face (the exterior side), as illustrated below.





Side view of front panel (viewed from right) showing the slight angle at the top, to match the slope of the side walls.

The angled top edge will result in the best fit, but you can just cut it square if your saw can't handle beveled cuts. If you cut it square, cut the piece according to the shorter **exterior** height. Using a square cut means that the back top edge won't quite align with the top edges of the side walls. But this whole area is covered by the lockbar when the machine is assembled, so it'll only be visible when you remove the lockbar to access the interior. The gap won't affect alignments for any of the trim hardware, so it won't have any functional impact.

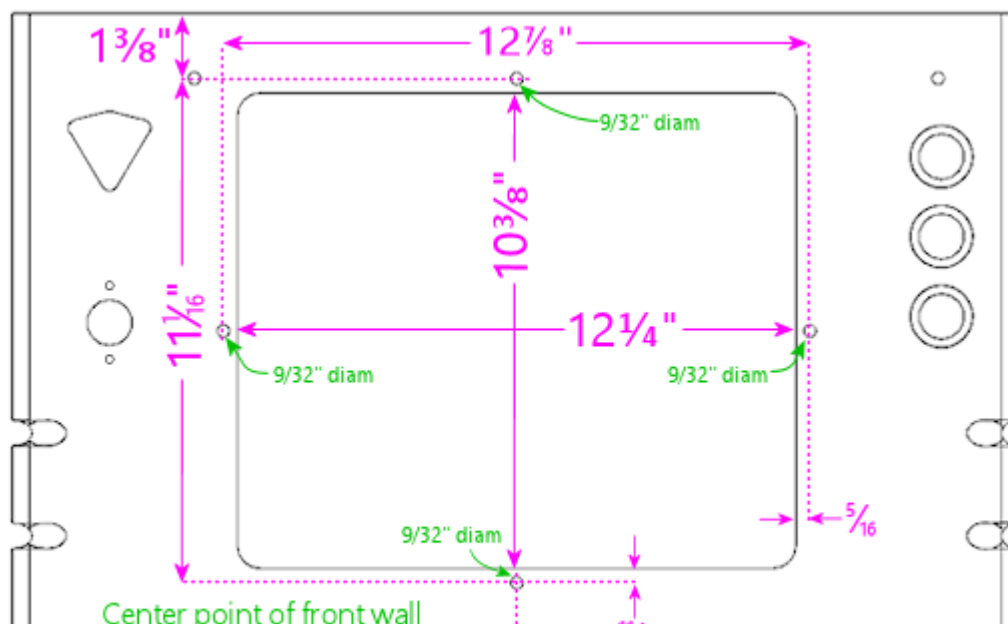
If you do use the sloping top edge, note that all of the measurements shown in our diagrams and plans are based on the **front** face - the shorter exterior side. Things will be off by $\frac{1}{8}$ " if you measure with reference to the top edge of the back side, since it's slightly taller. So be sure to do all of your measuring and drilling from the front side.

Edge finishes

The original WPC cabinets use a slight chamfer (a 45° bevel) on the outside bottom edge of the front wall, to soften the edge and reduce splintering. This is optional, but it's a little nicer than a sharp square plywood edge. See Edge finishes above.

Coin door cutout

The rectangular cutout in the center of the front wall is for a standard pinball coin door. All pinball manufacturers have been using the same coin door dimensions since the 1980s, so just about any coin door made for any modern pinball brand should fit this template, and you can also still buy the original Entropy coin doors that Williams used in their 1980s machines. I'm not as certain that pre-1980s doors are as standardized, so if you have an older coin door, you should measure it before using the diagram below, to make sure it fits.



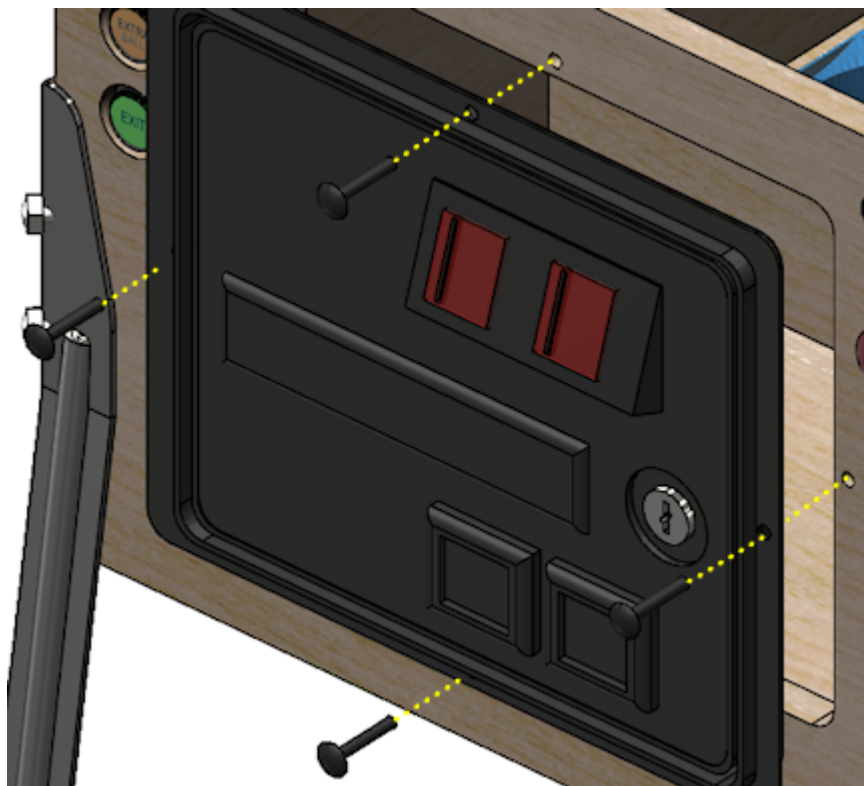


*Coin door cutout and bolt locations, viewed from the **interior** face of the front panel. Important: the measurements referenced to the top edge will be slightly different (about 1/8" less) when measured on the exterior face, because of the angled cut on the top edge. The interior face is about 1/8" taller than the exterior face because of the slant.*

The cutout in the diagram above is fairly generous. It leaves about 1/8" of play on all sides for an easy fit, so it should easily accommodate manufacturing variations in the coin doors. By the same token, when cutting it out, err on the side of cutting inside the lines. The actual cutout doesn't need to be even a hair bigger than depicted.

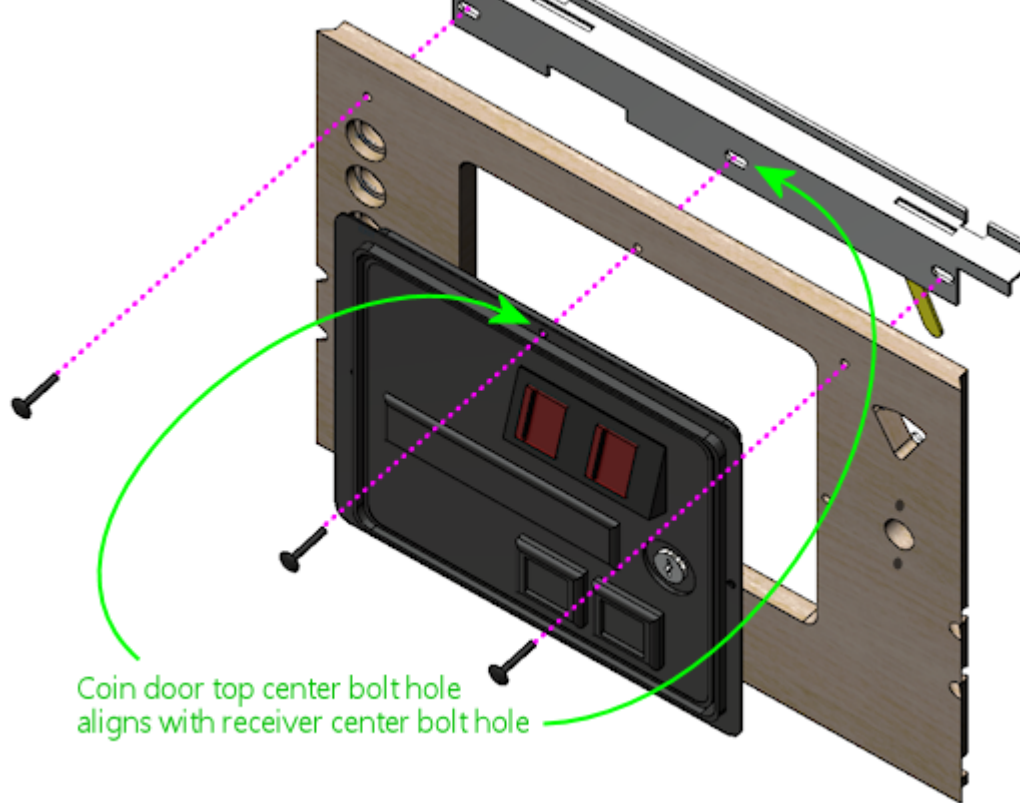
The four 9/32"-diameter drill holes around the perimeter of the coin door cutout are for the carriage bolts that fasten the door to the plywood. Use 1/4"-20 x 1 1/2" carriage bolts for these. Mate them to 1/4"-20 hex nuts, which go on the inside. The carriage bolts are available in black, which is what the WPC machines use to match the powder black finish of the WPC-style doors. The bolts are also available in stainless steel, chrome-plated steel, and silicon bronze, one of which might look nicer if you have a door with a metallic finish.

Note! The spacing between the coin door cutout and the four bolt holes around the perimeter is tight (only about 3/16"). Measure and drill carefully.



The coin door is usually centered left-to-right. If you're using a custom width, simply figure the position so that it's centered horizontally. **Don't** try to center it vertically, though. The vertical position has to align with your lockbar receiver, because the coin door's top center bolt hole has to align with the receiver's center bolt hole, as illustrated below.



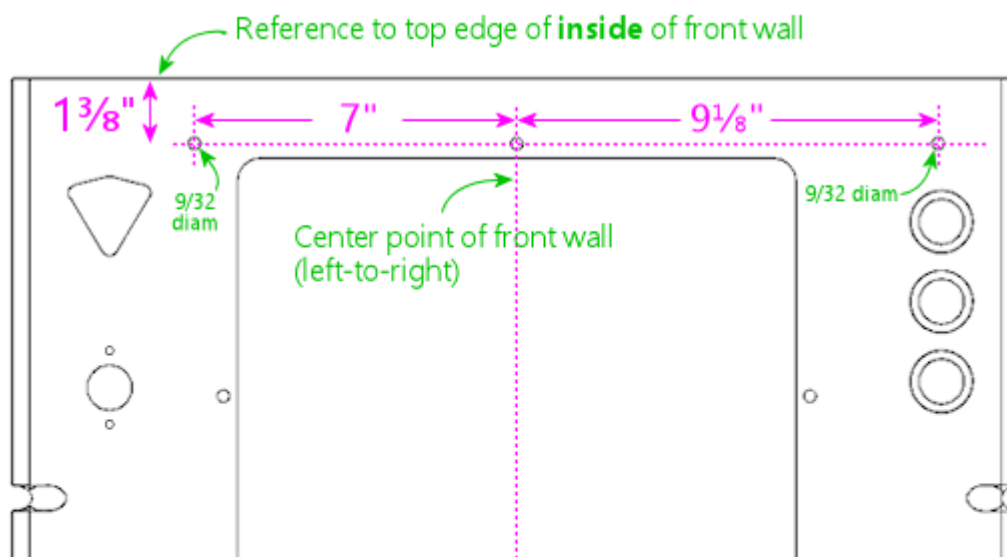


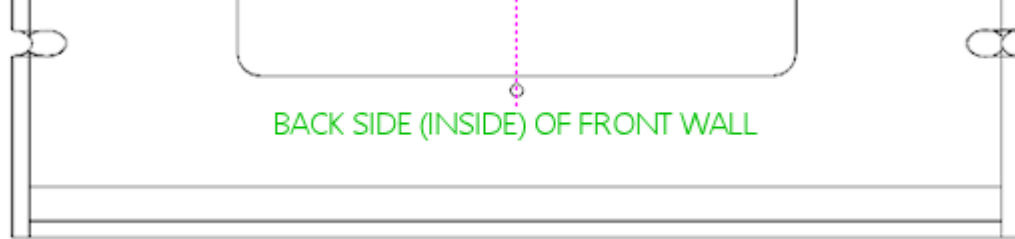
If you're using the standard WPC-era parts (a 1990s coin door and a Williams WPC lockbar receiver), the vertical position shown in our diagrams should align the receiver properly. If you're using different parts, they might have a different design, so you might need to adjust the vertical position to match. See the "dry fit" procedure in the lockbar receiver section below for advice on how to figure the right position for different parts.

If you're not using a coin door at all, you should obviously omit the rectangular cutout, as well as the drill holes around the perimeter. Note that the top center hole is shared by the coin door and lockbar receiver, though, so if you're using a standard lockbar receiver, you'll still need to drill that hole even though you don't need it for the coin door.

Lockbar receiver

The three small drill holes shown at the top of the front wall plan are for the carriage bolts that fasten the lockbar receiver to the front wall. (If you're not sure what the receiver is or what it's for, we'll explain more about it shortly.)





As with the coin door, use the center point of the front wall (left to right) as the horizontal reference point for the center hole.

The receiver has to be positioned vertically so that the lockbar will fit properly when inserted into the receiver. The vertical position of the bolt holes in our plans is specifically for a Williams WPC lockbar receiver, to place it at the right height so that the lockbar will fit properly.

Fine-tuning: I've found in my builds that the bolt positions above, which come from measuring original Williams equipment, can make the lockbar fit a little tighter than I like - not so much that it doesn't fit, just enough to make the fit feel a little clunky. Moving the bolt drill positions upwards by $1/32"$ to $1/16"$ might actually work a little better. If you do this, it would be a good idea to also move all of the coin door positions (cutout and bolts) by the same amount, since the coin door has to align with the lockbar center bolt. If you want to evaluate for yourself whether this would be a good idea or not, try the "dry fit" procedure described below - that's intended to help you re-figure the drill positions if you're using non-standard equipment, but it works equally well with the standard parts. I'd start by marking the standard positions above on your front wall panel, then do a dry fit and check the receiver's bolt holes against the markings. If they look a little off vertically, adjust accordingly. For the standard Williams receiver, the ideal vertical position is where the two little tabs sticking up at the front of the receiver line up exactly with the top edge of the front cabinet wall. (On my original Williams equipment, I see variations in this fit from the tabs being perfectly flush with the top of the wall, to being about $1/16"$ below.)

If you're using a WPC receiver, but other side rails or glass guides: The measurements here assume that you're using the WPC lockbar receiver **and** the WPC-style side rails and plastic glass guides. The thickness of the rails and guides is important to the overall positioning. If **any** of this is different in your setup, you might have to adjust the bolt positions vertically, since the lockbar might sit at a slightly different height than with the full set of standard parts. It's difficult to figure the right position on paper, because the parts have to fit together in a sort of 3D puzzle, and they fit tightly enough that there's not much room for error. I think it's easier and more reliable to do a "dry fit" with all of the parts together and take measurements from that. See the procedure below.

If you have to adjust the vertical position of the bolt holes, you should adjust the coin door position to match. The coin door has to line up with the lockbar's center bolt, so if you move the lockbar bolts up or down, the coin door has to be moved up or down by the same amount.

If you're not using a WPC receiver: There are several other options for a lockbar receiver besides the WPC part. If you're using something different, it'll probably have a whole different drilling pattern for its fasteners. It might not even use the same bolts. The best way to figure the right drilling positions (if needed at all) is to gather your equipment and do a "dry fit" as described below.

Note that the center hole is still needed for the coin door, if you're using one, regardless of whether your lockbar uses it.

Dry fit: Here's a procedure you can use to fit your lockbar and other related parts together prior to drilling any holes, to measure or fine-tune the positioning:

- Set up the front and side walls in their assembled positions.
- Set up the side rails with the glass guides, if they'll be part of the final setup.
- Plug the lockbar into the receiver.
- Position the lockbar at the top front where it'll be during normal use. You want the lockbar to sit snugly on top of the side rails when everything is put together, so at this stage it's a good simulation to simply set the lockbar on top of the rails.
- Now hold the receiver's front surface flush against the inside front wall. Make sure the lockbar is still where you want it.
- Mark the positions on the inside of the front wall corresponding to the positions of the three bolt holes in the receiver. (The bolt holes in the receiver are actually little slots, to give you a little wiggle room to make up for measuring errors, so mark the position at the center of each slot.)

You can now take it all back apart, and drill at the marked positions instead of the ones in the plans.

How the lockbar works

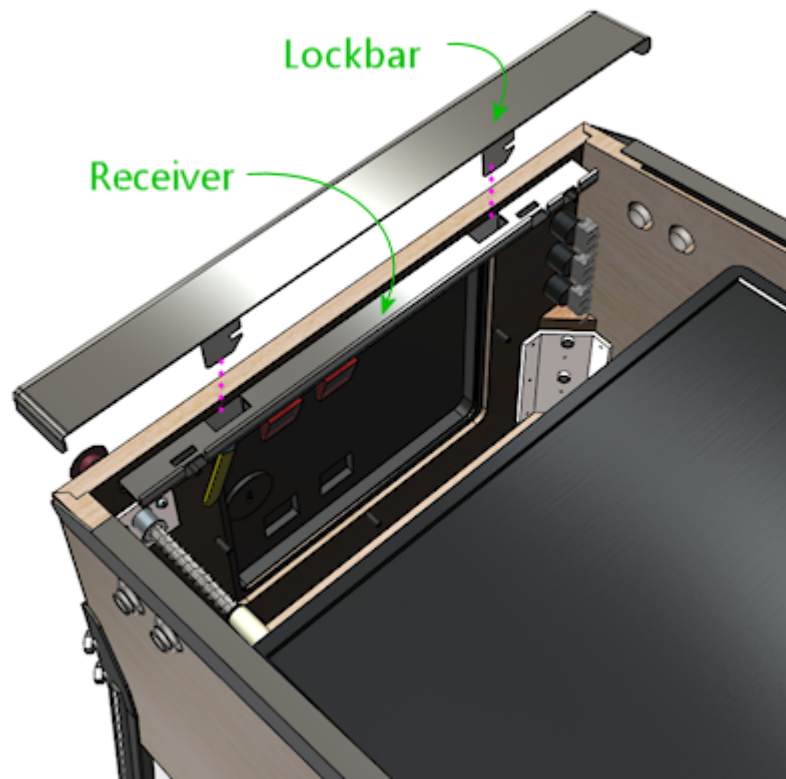
In case you're not already familiar with how all of the pinball trim pieces work, here's a brief overview.

The "lockbar" (also known as the "lockdown bar") is the metal trim piece along the top front edge of the machine. It's so named because it serves to lock the top glass cover in place. It also functions as a trim piece, for the sake of appearance as well as to provide a comfortable place to rest your hands while operating the flipper buttons. Standard lockbars have nice smoothly rounded corners. Try playing a round on a machine with the lockbar removed if you want experience for yourself how unpleasant the plywood edges are as a hand-rest.



If you're using standard pinball parts, the lockbar mates with a part inside the

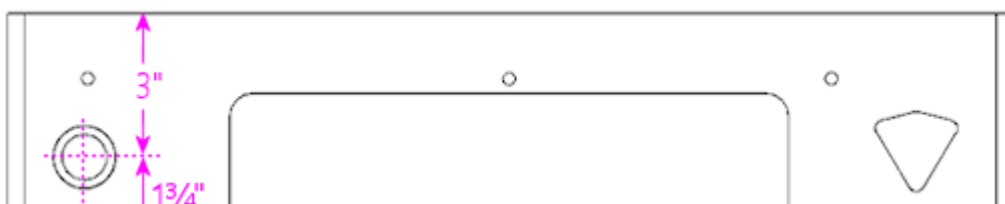
cabinet called the "receiver". A couple of prongs that stick down out of the lockbar fit into receptacles in the receiver, where there are some spring-loaded latches that grab the prongs and secure the lockbar. A lever on the receiver, which you can reach through the coin door opening, lets you release the latches and free the lockbar. With the lockbar off, you can slide out the glass to access the interior. It's all cleverly designed to let an operator open up the machine quickly and without any tools, while keeping it buttoned up against intrusion by mischief-makers.

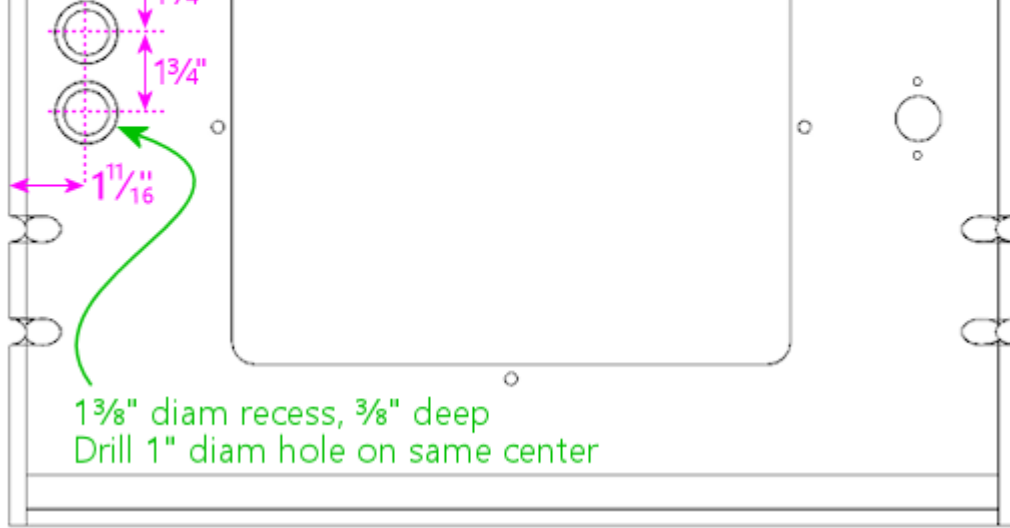


The receiver attaches to the inside of the front wall of the cabinet. It's fastened with three carriage bolts. This is what the drill holes at the top of the front wall are for. The center bolt is shared between the coin door and receiver - both parts have holes in this position that align when everything is assembled. This is why the vertical position of the coin door is so important: the coin door aligns with the lockbar receiver, and the receiver has to align with the top of the wall so that the lockbar fits properly.

Front panel buttons

The three large circular holes at the top left of the front panel diagram are for buttons that the player uses to start and exit games and otherwise interact with the software. Our plans assume that you're using SuzoHapp small pushbutton (pictured at right), which are the type used for most of the front-panel button on real machines since the 1990s. These are the exact type that most pinball suppliers will sell you if you buy a replacement Start button, Extra Ball button, or generic "pushbutton with lamp assembly". There are other similar buttons available from other companies that you can use as well, but you might need to adjust the drilling dimensions and/or spacing for other models.



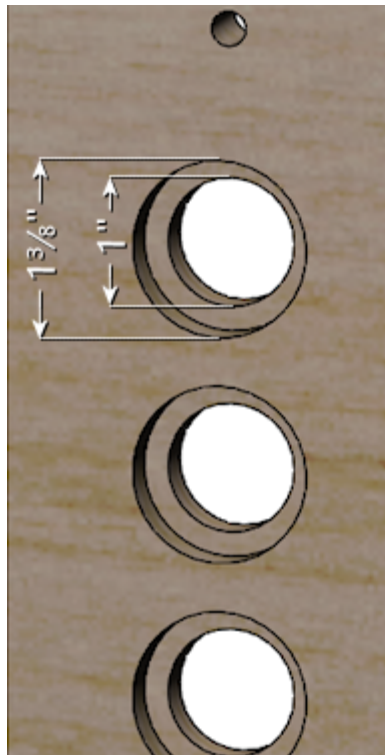


Typical button positioning, with three buttons (usually "Start", "Extra Ball", and "Exit"). Viewed from front/outside of front panel.



Attention: Replacement cabinet builders: If you're building a replacement cabinet for a real pinball machine, take measurements from a factory original of the same title to determine the button placement. The cabinet artwork for some titles includes designs around the front panel button(s), so you might also want to check alignment with the art.

For the SuzoHapp style of pushbuttons, drill the holes in two stages. First, using a $1\frac{3}{8}$ " Forstner bit, from the front (outside) face, drill a recess $\frac{3}{8}$ " deep (about half of the plywood thickness). **Don't** drill all the way through. In the diagram, the recess is the larger circle drawn around each button. Then drill a 1" hole on the same center the rest of the way through. The recess allows the button to sit flush with the front surface of the cabinet.



Above left: Drilling detail for the button holes, viewed from the exterior face. Drill a $1\frac{3}{8}$ "-diameter recess to $\frac{3}{8}$ " depth (about halfway through the plywood). A Forstner bit works best for this. Then drill a 1" diameter the rest of the wall

through, on the same center. Above right: when installed, the buttons are recessed in the routed depressions, so the button faces are roughly flush with the outer surface of the cabinet.

The recess is optional. It's the way that the buttons were mounted on the real 1990s machines, and I think it looks more finished. But if you want to keep things simpler, you can skip the recess and simply drill a 1" hole straight through. The buttons will jut out by about a quarter inch if you omit the inset, but this won't look "wrong", since the buttons are trimmed to work with this mounting style as well.

Our plans show the positions for three buttons, but that's only a suggestion. As far as software usability goes, the virtual pinball software more or less requires a minimum of two buttons: "Start" and "Exit". The third button in our plans can be assigned any other function of your choice, or you can omit it entirely. See Chapter 34, Cabinet Buttons for ideas and recommendations. I think it's a good idea to include a third button, even if you don't already have a clear use for it in mind, since it will be hard to add one later. You can change the meanings of the buttons at any time in the software, so you're not stuck with the functions you choose initially. I assigned my third button as "Extra Ball", since that's used on a lot of real machines from the 1990s. Another useful function is "Coin In" (to simulate inserting a coin), although I prefer implementing that via the coin return buttons on the coin door, since that's a more natural and inconspicuous place for it. Other possibilities include game-specific extra buttons, or special functions in your game navigator software.

Omitting a button is easy. If you only want to include two buttons, simply drill the top two holes at the positions shown, and skip the bottom one.

It's difficult to add more buttons beyond the three shown given the space constraints. With the spacing shown, there's not enough room for a fourth button at the top, since the lockbar receiver will get in the way, nor at the bottom, where the leg brackets will conflict. However, you just barely make room if you move the top button up about 1/2" (that's the limit before it conflicts with the lockbar receiver) and then tighten up the spacing on the other buttons by about 1/8". That will give you just enough room for a fourth button at the bottom.

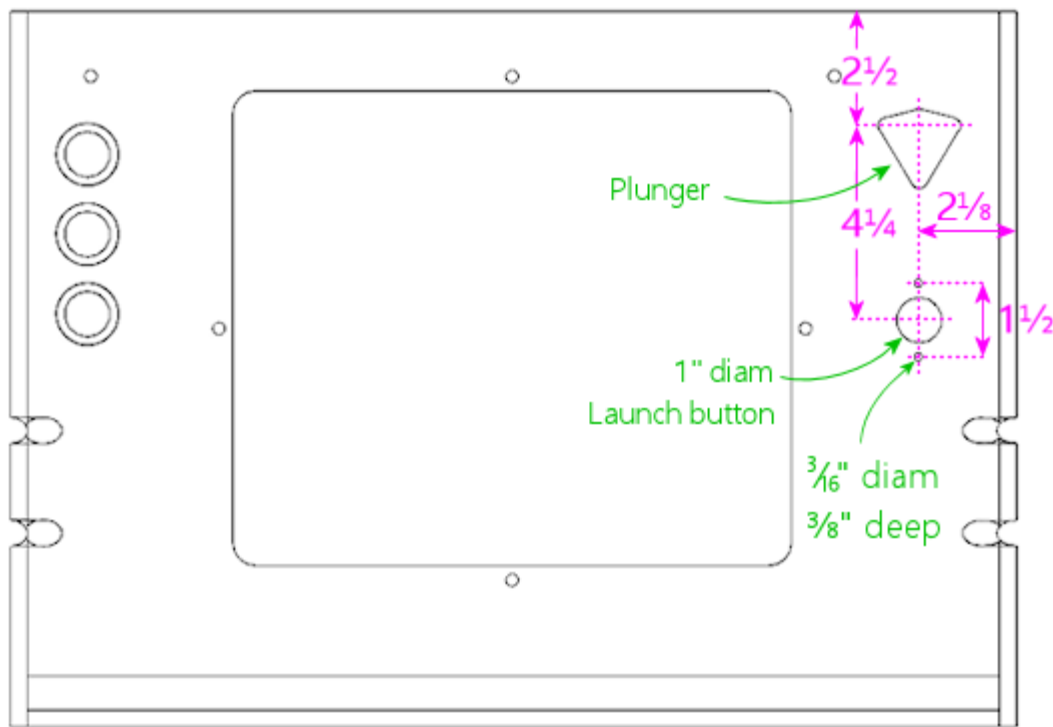
Plunger and Launch button

Our plan includes a traditional mechanical plunger, at the standard position used on nearly all real machines, at the upper right corner of the front face. We also include a Launch Ball button, situated just below the plunger, to accommodate tables that originally used a button or trigger in place of the traditional plunger.

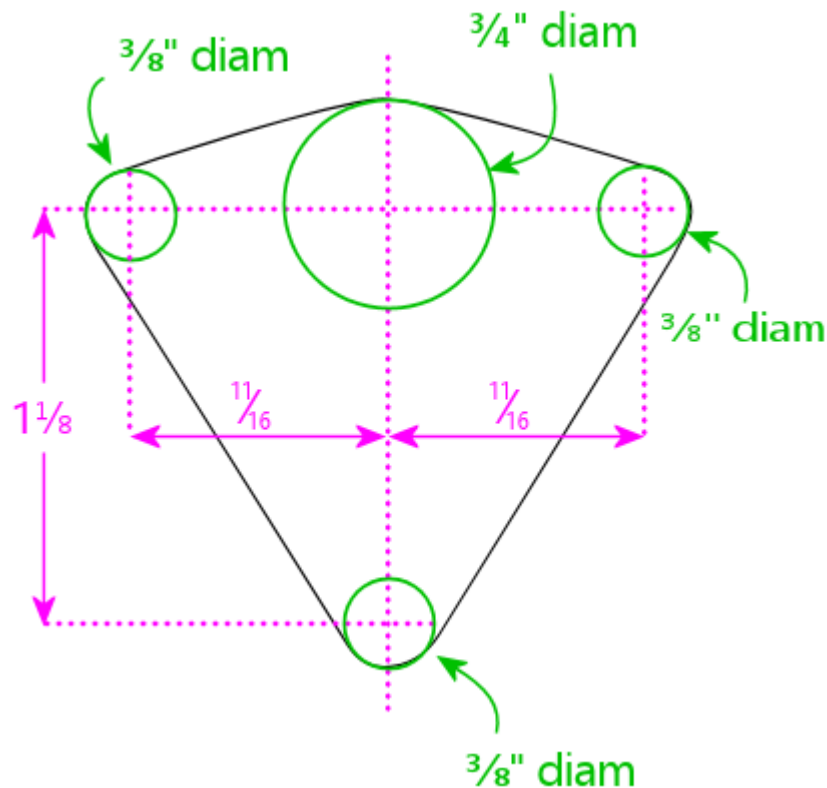
Be aware that this traditional plunger position doesn't work for everyone! In particular, it can get in the way of the TV if you want to place the TV very close to the front wall. See "Other plunger/Launch button layouts" below for an alternative plan that swaps the positions of the plunger and launch button to make room for the TV. If you haven't thought about the TV conflict issue, see "The dreaded plunger space conflict" in Chapter 29, Playfield TV Mounting and "Positioning the plunger" in Chapter 37, Plunger.



Attention: Replacement cabinet builders: If you're building a replacement cabinet for a real pinball machine, don't rely on my plunger positioning! The plunger on a real machine has to line up with the shooter lane on the playfield, so it depends on the playfield depth, which varies from one title to the next. The differences can be substantial - the Williams System 11 games generally have the plunger about 1" higher than on the WPC games. Your best bet is to measure the factory drill positions from an original cabinet for your specific game. If you don't have access to one, try asking on a pinball owners forum such as Pinside.



Drilling positions for plunger and Launch Ball button, with the plunger in the standard position used on real machines, and the Launch button below.



Drilling pattern for the plunger opening. Reference the vertical location from the main plan to the top dotted line. For the standard plunger-on-top configuration, this is 2 1/2" from the top of the panel.

To cut the plunger opening:

- Drill a 3/4" hole at the large green circle at top center. It's best to use a router, or a drill with a hole saw bit or Forstner bit. (Don't use a spade bit; they make ragged, chipped holes in plywood.)

- Drill $\frac{3}{8}$ " holes at the three smaller green circles.
- Use a jigsaw or router to cut along the perimeter of the shape described by the four holes, shown as the black outline on the diagram.

The illustration at right shows how this looks when assembled.

This arrangement, with the plunger on top and a Launch button below, is the one I prefer. It has two main virtues. First, the plunger position matches the real machines of the 1990s, so it looks "normal" if you're used to the way those machines look. Second, it's nice to have the dedicated Launch button for tables that use one, and this placement looks the most natural to me. You won't actually find any real tables that have both a plunger and a Launch button, so that much is not quite authentic - but many real machines did have *some* sort of button at the same location (e.g., an Extra Ball button), so it doesn't look at all out of place.



Other plunger/Launch button layouts

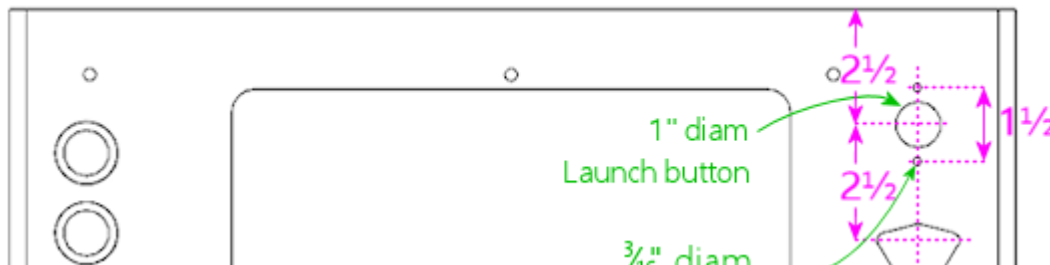
The traditional plunger location shown above doesn't work for everyone, because it can create a space conflict with the TV if you want to position the TV at the very front of the cabinet. Before you drill anything, take a moment to consider if you'd prefer some other setup. Here are the most common options:

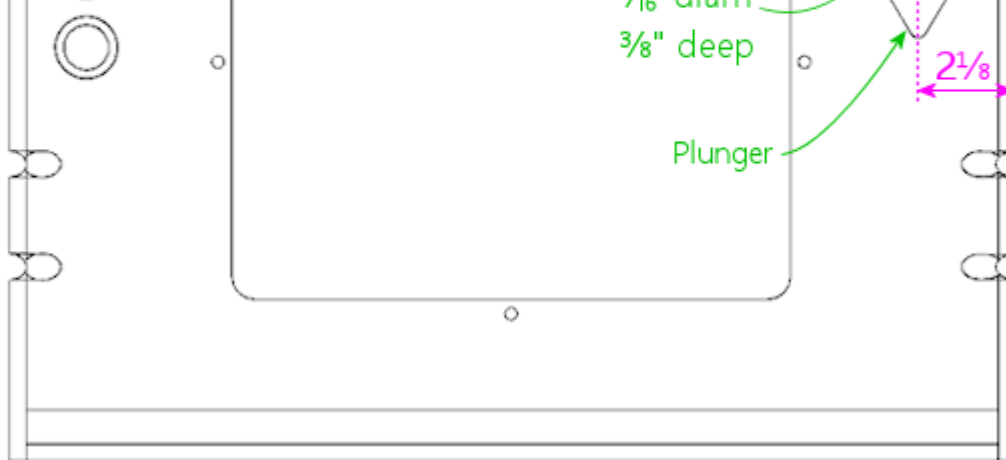
- Include only the plunger, with no Launch button. Some people prefer a more authentic-looking setup with just the plunger. This an easy modification: just don't drill the hole for the Launch button.
- Invert the arrangement so that the Launch button goes on top and the plunger goes below. Some people use this arrangement to make room for the TV to fit closer to the front of the cab. To make this change, use the inverted plan below.
- Include only the plunger, but lower it to get it out of the way of the TV, so that the TV can be mounted closer to the front of the cab. To implement this, use the inverted plan below, and skip drilling the hole for the Launch button.
- Include only the Launch button, with no plunger. To do this, use the inverted plan below, but don't cut the plunger opening.

For more advice on choosing among these options, see Chapter 37, Plunger.

Inverted plunger/Launch button

Here's the inverted layout, with the plunger below the Launch button. This places the Launch button at the exact position used on real machines that use this control (which also happens to be the standard plunger position, not surprisingly), so it'll look authentic as far as that goes; of course, the addition of the plunger below the button isn't to be found on any real machines.





Inverted arrangement with the Launch button on top and the plunger on the bottom.

Note that the spacing between the plunger and Launch button is a tiny bit tighter with the inverted layout than with the normal layout (by about an eighth of an inch). This is due to space constraints. The plunger can't safely be moved much lower, because the exterior side of the plunger housing will conflict with the front right leg if you do. If you really need to move the plunger even lower than shown (to make room for the TV, for example), you might be able to eke out a few extra 16ths of an inch, but it might be an uncomfortably tight fit. Measure your actual parts carefully before making changes.

Other cutouts

I don't recommend any other controls or ports in the front wall, since this is the most conspicuous part of the machine other than the playfield area, it's already pretty busy with just the standard controls. However, there are a few extra items that some people add here:

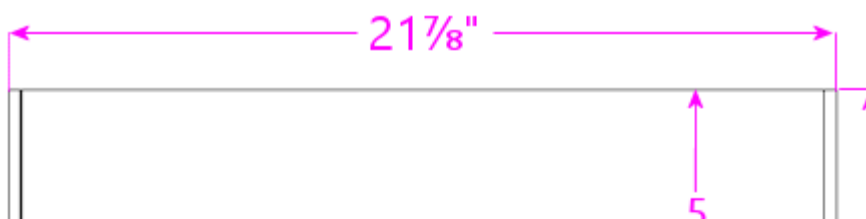
- Volume controls
- Night mode switch
- USB/keyboard/mouse ports

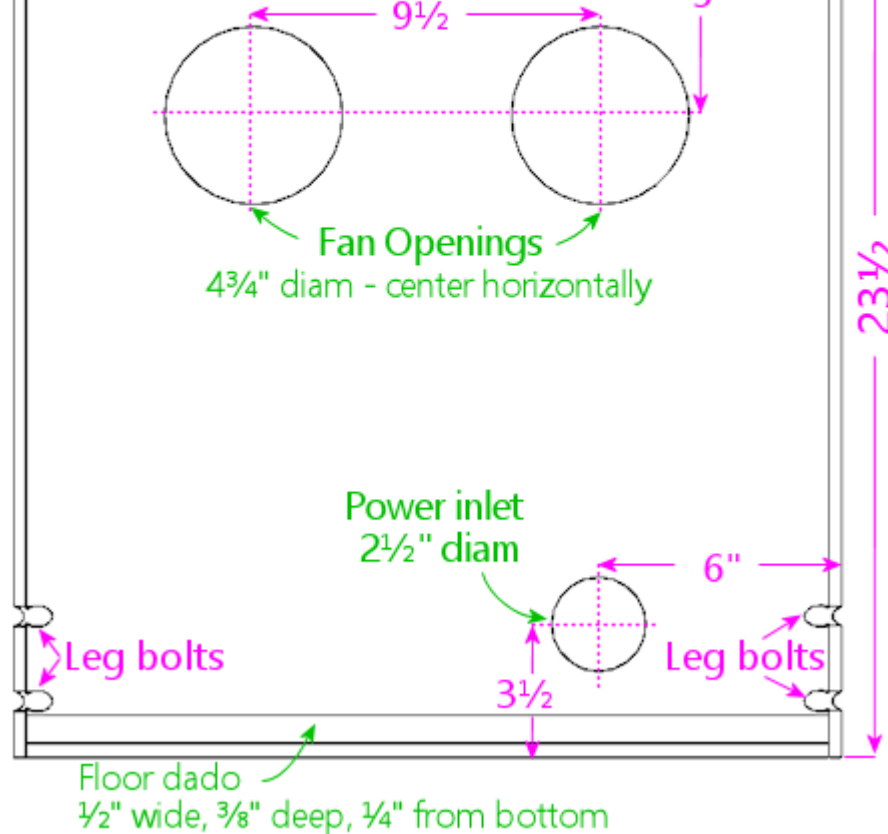
I'd personally avoid the front panel for all of these and place them on the bottom or back of the cab instead, where they'll be less visible.

For volume controls, I'd recommend using doubled-up flipper buttons instead of a separate knob (see my PinVol page for an explanation). But if you really want a separate knob, and you don't want to have to reach under the machine to operate it, one way to make it inconspicuous is to install it in the coin door, by drilling a hole for the knob stem.

Rear wall

The rear wall is a lot simpler than the front wall. It just has a couple of openings for cooling vents, and another for the power inlet. Nothing has to align with standard trim pieces here, so the placement of the openings is flexible.





Rear wall, viewed from the interior side.

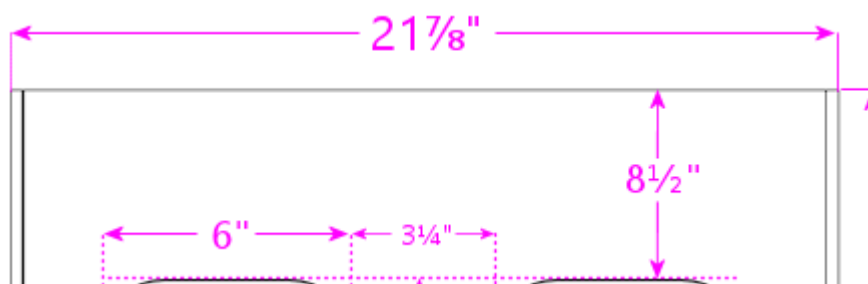
The fan openings are designed to accommodate 120mm PC case fans, mounted just behind the openings (on the inside of the cab) and oriented to blow air out the back. These aren't authentic to the original WPC design (for the original layout, see the diagram below). The WPC machines had smaller, passive vents. Most virtual cab builders want to include fans to actively blow air through the cabinet for cooling, which the larger openings accommodate.

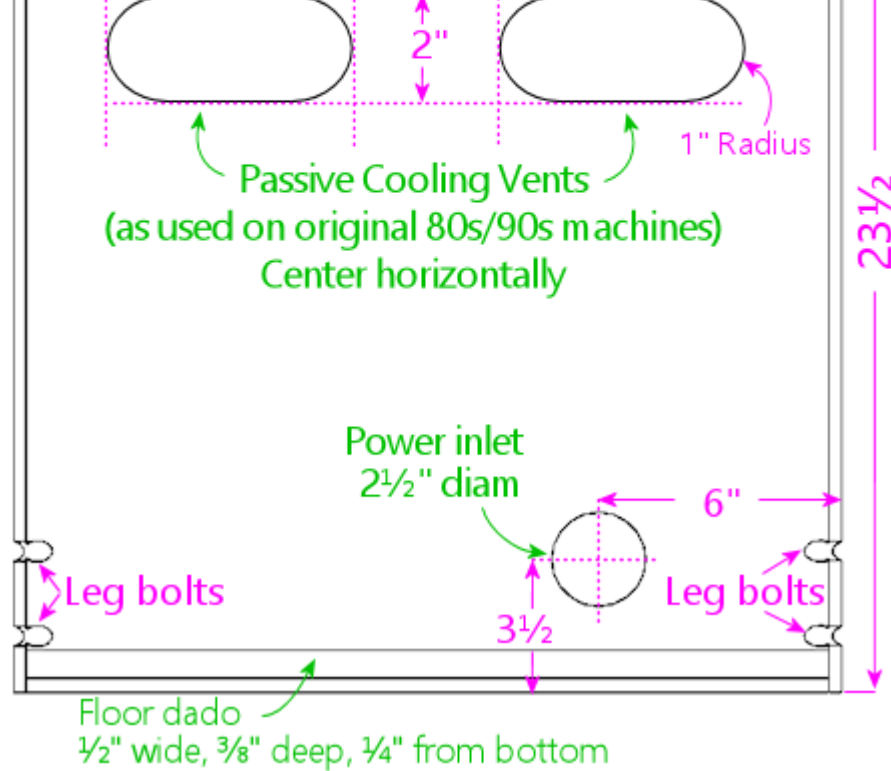
The power inlet opening is there to pass the machine's main power cord through the back, for plugging into a wall outlet. This is the same as the original WPC equipment, which has a C14 power inlet (the same type of power cord connector used on most desktop computers) behind the opening.

The size and placement of the fan openings and power inlet are merely suggestions. Customize them as you see fit. Take care that anything you install on the back wall doesn't get in the way of the playfield TV, but that usually isn't a problem, since the back end of the TV is usually well forward of the back wall. The leg notches and floor dado should be implemented as shown, since those do have to align with other parts.



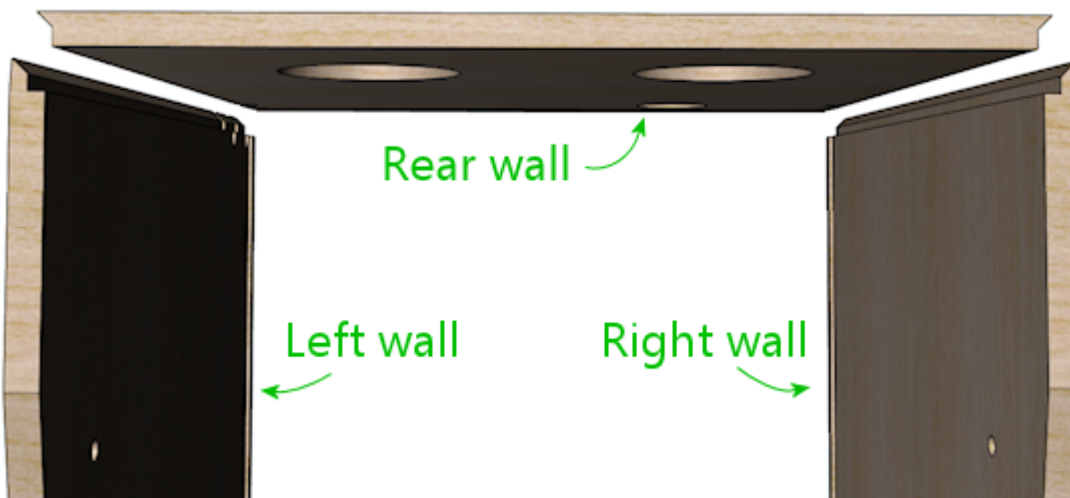
Attention: Replacement cabinet builders: The circular fan openings shown in the diagram above aren't authentic to the original WPC cabinet design. A replacement cab for a mechanical pinball machine should follow the pattern used in the Williams cabinets of the 1980s and 90s, with passive vent openings as shown below.





Original rear wall design used on the real machines, with passive cooling vents instead of fan openings, viewed from the interior side. This is the layout of the Williams cabinets of the 1980s and 1990s. Virtual cab builders usually replace the vent slots with larger circular openings that can accommodate PC case fans, to provide active cooling. Virtual cabs tend to need active cooling in the main cabinet because they typically house a TV and a PC motherboard, both of which can generate a lot of heat.

Most people use the same joinery style for the rear wall as for the front wall, but that's not required. I think a mitered joint (such as a mitered rabbet or lock miter) is nice here, since it yields seamless corners, but that's probably not as cosmetically important as it is at the front. A simpler joint that produces visible seams, such as a rabbet or even a butt join, can be perfectly adequate aesthetically.



Top view of rear section, showing the joinery shapes at the rear corners. This uses the mitered rabbet as described in the side walls section earlier.

The leg bolt notches work exactly like on the front and side panels. Use the same measurements as the **rear** leg notches on the side panels, since those need to align with the ones on the back wall when the cabinet is assembled. See the side wall section above for details.

The floor dado is a routed groove that the floor fits into when you assemble the cabinet. This is the same as the floor dados on all of the other pieces: use a 1/2" straight bit to route a groove 3/8" deep (about half the thickness of the plywood), parallel to the bottom edge, offset 1/4" from the bottom edge. See the side wall section above for a diagram. As mentioned earlier, some other published WPC plans offset the dado by 3/8" from the bottom rather than 1/4", which might be preferable for added strength at the joint. Whatever offset you choose, use it consistently for all of the floor dados on the sides, front, and back.

Power inlet

The hole near the lower right of the back wall plan is for the main AC power inlet. On the real machines, this is a 2 1/2" diameter hole positioned as shown. There's nothing special about this location for a virtual cab; move it and/or resize it as needed for your own power supply setup. If you're not sure how you're going to set up the main power supply, you can just follow the generic plan, since it's pretty versatile; the opening is large enough that you could just feed a power strip's cord or an extension cord through it, and it could also accommodate a C14 inlet mounted in the opening. You can drill a hole of this size with a hole saw bit, or using a hand router with a circle jig.

Fan openings

The fan holes in our back wall plan represent a deviation from the real WPC cabinet design, to meet the special needs of the virtual cab. Real pinball machines don't need much cooling for the main cabinet, so the WPC cabs just have a pair of small passive vents at the back. Virtual cabs, in contrast, tend to need active cooling with fans, since the main cab has a big TV and (in most cases) a PC motherboard.

Our plans provide two openings in the rear wall designed for PC case fans. The idea is that you place an exhaust fan (blowing air out of the cabinet) on the inside of each opening. The cabinet floor (which we'll get to next) has another similar opening for an intake fan. This arrangement is designed to work with the natural air flow from the tilt of the monitor: the tilt makes the monitor higher at the back, so warm air will tend to flow towards the back of the cabinet as it rises. The exhaust fans at the back will help remove the hot air and pull cooler outside air into the cabinet from the floor vent.

The holes shown in the diagram are for 120mm fans (about 4 3/4" inches diameter). This is a common size for PC case fans, but other sizes are available; some people like to super-size their fans because larger tends to be quieter. Resize the openings for your fans as needed.

There's nothing magical about our placement of the fan openings, so move them as needed. I recommend keeping them relatively high up on the wall to take advantage of the natural flow of rising warm air. The point is to remove the hottest air from the cabinet, and that will tend to move towards the upper portion of the space.

For more on cooling, see Chapter 28, Cooling Fans.

Other rear wall cutouts

Here are some other optional items that you might want to consider, as long as you're drilling holes in this piece. There's no standard placement for any of these, so use whatever location is convenient for your setup.

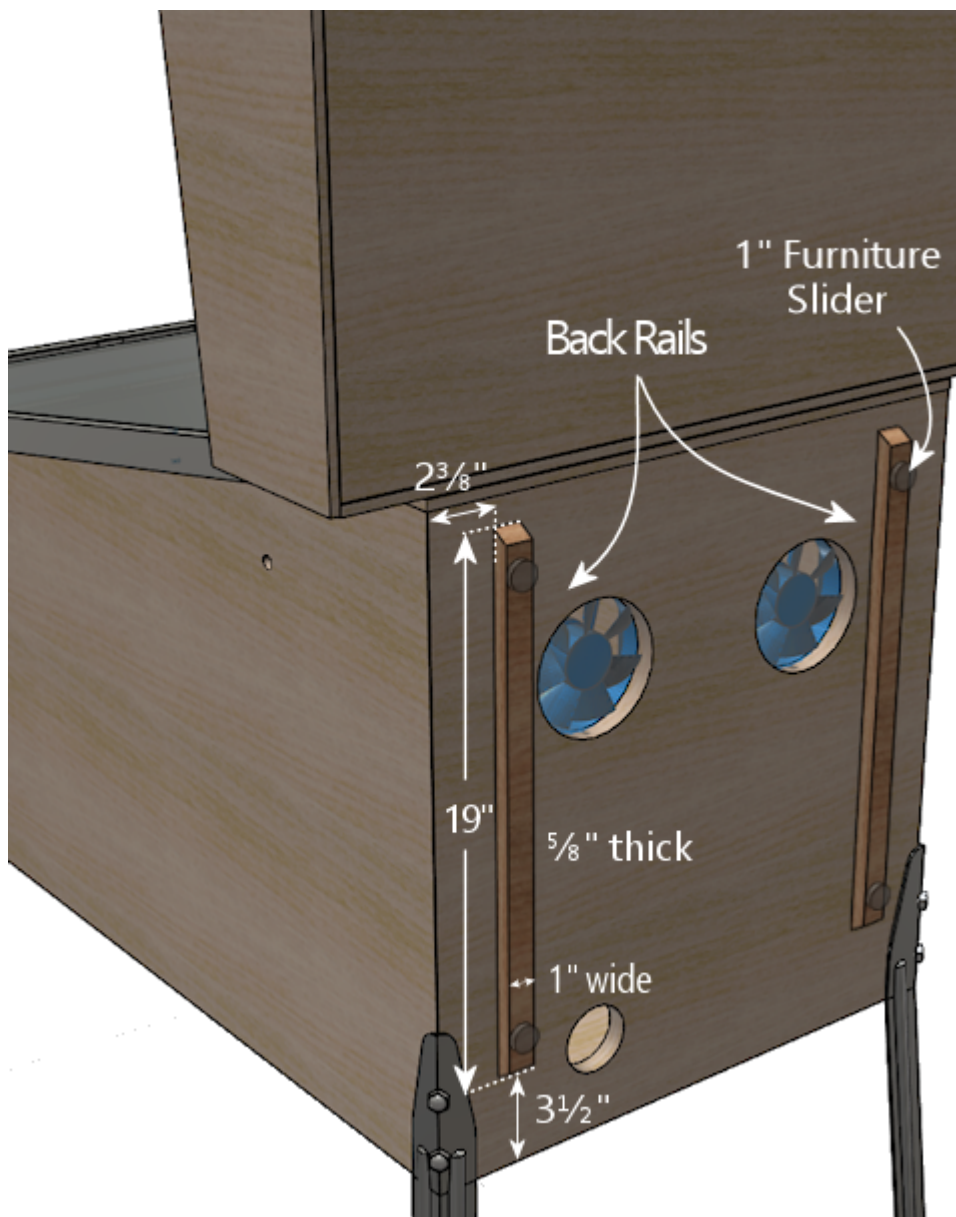
- Ethernet port. Wired network ports can come in handy even if you're planning to install a Wi-Fi card or powerline Ethernet. The rear of the cabinet adjacent to the power inlet is an excellent place for this. Keystone jacks are useful here. See "External I/O plugs" in Chapter 27, Installing the PC.

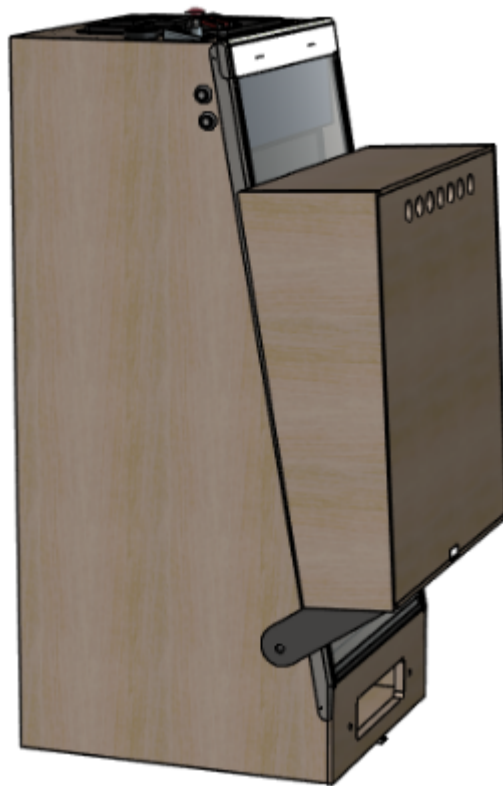
- USB ports. It's also good to have some external USB ports, and the back of the cab makes a convenient place for a couple of these. As with Ethernet, you can use Keystone jacks. If you're installing a Keystone jack plate for Ethernet anyway, you can make it a 3-gang or 4-gang plate and populate it with a couple of USB ports while you're at it.
- Keyboard/mouse ports (these are usually just more USB ports). I prefer the floor of the cab near the front, since that's where you'll actually want to use the keyboard and mouse, but the back of the cab will do if you just want a single cluster of ports.
- Openings to pass wires for light strips on the back of the cab (see Chapter 58, Undercab Lighting)

Back rails

The real WPC cabinets have a pair of wood rails on the back, as illustrated below. Each rail has a pair of hard plastic furniture slider pads attached (the nail-in type, typically 3/4" diameter, white or tan), one at each end. These are designed to let you stand the machine on its back, with the backbox folded. The machine is more compact in this configuration, which can be helpful for moving, shipping, and storage.

These rails are optional. If you want to include them, cut the two strips at the size shown. On the WPC machines, the ends are beveled at about 30°.



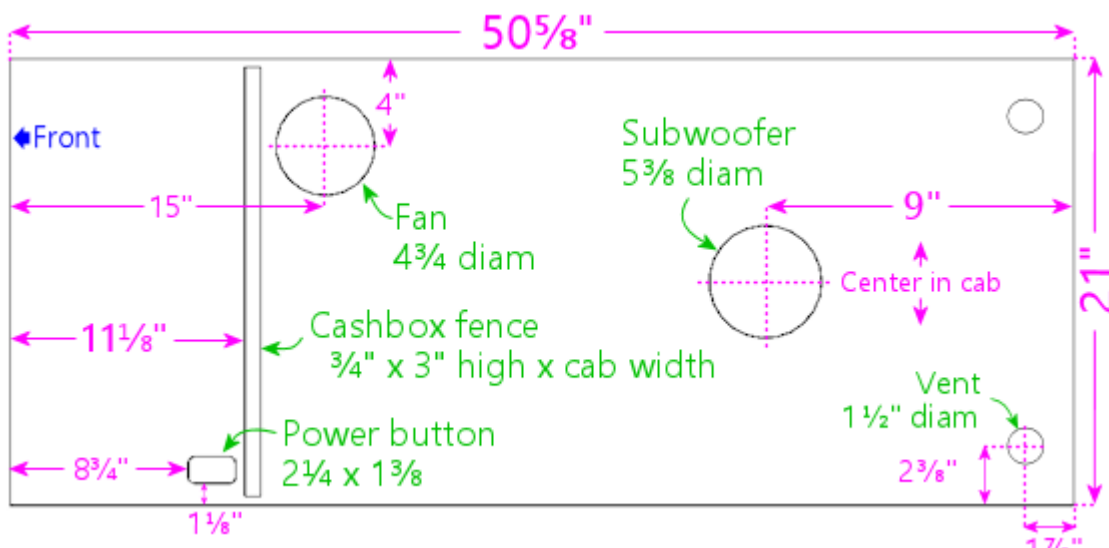


Shipping configuration: legs removed, backbox folded down, placed on back. The machine can be strapped to a pallet and boxed or plastic-wrapped. This is good for freight shipping because it has relatively small footprint and it's easy to move with a pallet jack.

Floor

The floor is constructed from 1/2" thickness plywood (or particle board or MDF, if you prefer).

Our plan for the floor of the cab makes some concessions to the special needs of the virtual cab, so its cutouts aren't quite identical to the normal WPC floor design. In particular, we moved the subwoofer from roughly the middle to closer to the rear of the cab, and we added an opening near the front for a PC case fan to actively draw outside air into the cabinet, to supplement the fans at the back that blow hot air out. The power button cutout is also slightly wider than on the WPC machines (1-3/8" in this plan vs. 1-1/8" in the originals), to accommodate an arcade-style pushbutton.



When you assemble the cabinet, the floor fits into grooves (dados) routed in the side, front, and back walls of the cabinet. No additional joinery routing is required on the floor piece itself.

The "cashbox fence" isn't a cutout - it simply marks the location of a short wall installed here on the real machines, mostly to hold the cashbox in place. (The cashbox is a plastic box that sits under the coin slots to collect the booty. It comes in a standard size for Williams machines; you can buy one from a pinball vendor.) If you're not planning to use the standard type of cashbox, you can omit the fence, which will leave more open space for PC parts. However, you'll certainly need *some* sort of container to collect coins, if you're using them; you don't want loose metal discs rolling around your electronics-packed cab interior. The standard cashbox is a convenient solution. But it's also awfully large. On my own cab, I improvised a much more compact coin box using a plastic food container.

If you want to install the fence, it's 3" tall by $\frac{3}{4}$ " thick. Cut the length to match the inside cabinet width. Mount it at the position shown (or whatever position is right for your cashbox, if you use something custom). This isn't a structural element, so it doesn't have to be very strong; you can fasten it with glue and/or nails. Note: the distance shown ($11\frac{1}{8}$ ") is from the front of the floor piece, which recesses into the dado in the front wall by about $\frac{3}{8}$ ". If you install this after assembling the rest of the cab, it goes $10\frac{3}{4}$ " from the inside front wall.

The subwoofer opening is shown at the size used in the WPC machines, but the position is further back than in the real machines, where it's closer to the middle ($22\text{-}1\frac{1}{4}$ " from the back, to be precise). The virtual plan moves it back to create more contiguous floor space for the PC motherboard. I don't think it'll affect the acoustics much (if at all) if you want to move it further back still, for an even bigger stretch of open space.

You should consider changing the diameter of the subwoofer cutout to match the speaker you select. The $5\text{-}3\frac{3}{8}$ " diameter cutout is based on the 6" speakers used in the WPC machines. Those are small by modern standards; automotive subwoofers are in the 8"-and-up range. If you do use a larger speaker, it'll sound better if the opening is roughly the same size as the speaker aperture.

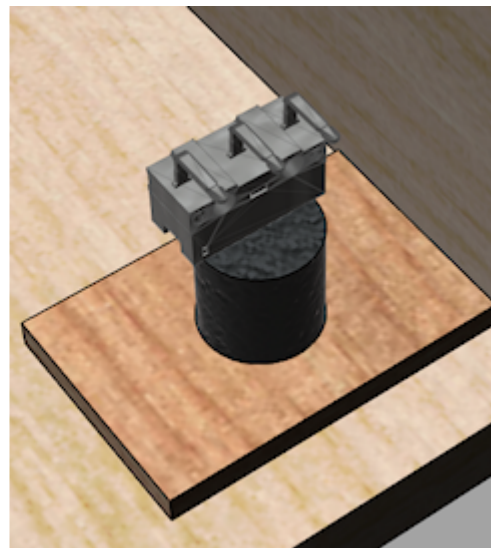
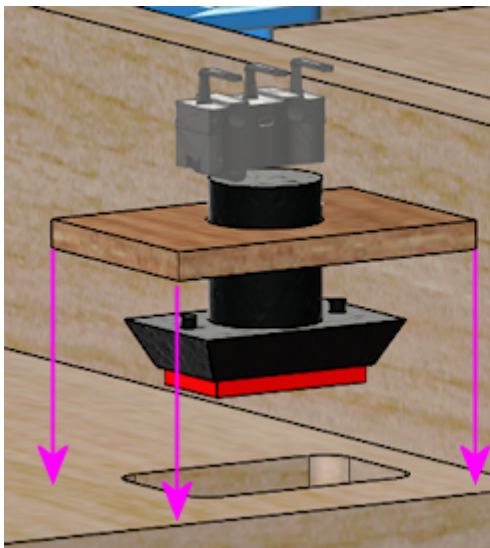
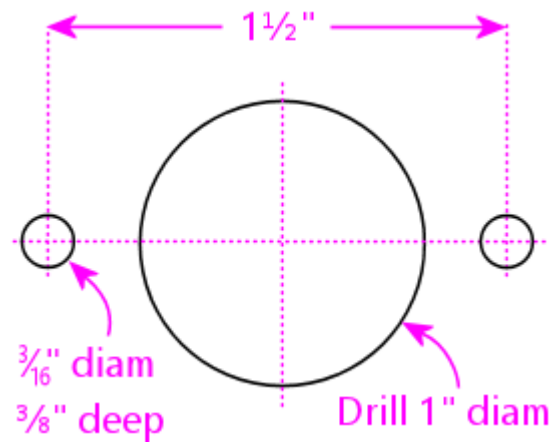
The large fan opening towards the front isn't part of the original WPC design. It's a virtual cabinet add-on for our greater cooling needs. This is meant to be an **intake** fan, with a PC case fan mounted on the interior surface and oriented so that it blows air **into** the cab. This helps draw in cool air from the bottom to replace hot air being blown out by the fans at the back. See Chapter 28, Cooling Fans for more on this subject.

As with the fan openings in the back wall, the position and size shown are only suggestions, and there's nothing special about the exact placement shown, other than that it's generally close to the front of the cabinet to promote front-to-back air flow. The opening is sized for a common 120mm PC case fan. Some people think it's better to use two intakes to match the two vents in the back, so you could add a mirror-image opening on the opposite side (near the power button). But I wouldn't go too overboard on adding fan vents, as they eat into the space available for the PC components and other items, plus too many cutouts will weaken the floor.

The two small ($1\frac{1}{2}$ " diameter) holes near the back corners are from the the original WPC design, and they're for ventilation. These are redundant in our design with the added opening for the PC intake fan, but I'd keep them anyway for freer air flow.

They don't take up much floor space.

The power button opening is shown at the standard position for real machines, which works equally well in a virtual cabinet. The cutout in our plan is slightly wider than in the original WPC design (1-3/8" vs. 1-1/8"), to accommodate more types of buttons. The real machines use a "hard" on/off switch here that controls the AC power to the main transformer, so turning it off is basically the same as unplugging the machine from the wall. On a virtual cab, we usually want a "soft" power button instead, since we're working with a Windows PC, and Windows doesn't like abrupt power loss. Windows wants the power to remain on throughout the shutdown process, so a soft power control is needed. You just need a pushbutton that's wired to the "power button" connector on the PC motherboard. I use one of the common SuzoHapp rectangular arcade-style pushbuttons. This type of switch can be mounted as illustrated below, which recesses it nicely into the opening.



Installing a SuzoHapp rectangular pushbutton (part #D54-0004-5x) in the power button opening. Cut a small piece of plywood (about 2" x 3") to serve as the mounting plate. Drill holes as shown. Mount the button on the plate, then insert the button into the cutout. Attach the mounting plate to the cab floor with a couple of small wood screws.

Cab floor materials

The real WPC machines had particle board floors. I'm usually all for faithful replication of the originals, but this is a detail that I only see as a negative. I'm sure the only reason they used particle board is that it cut a few dollars off the cost. Plywood is lighter and stronger, so I'd stick with that. The problem with particle

board is that it tends to sag over time, especially in a big unsupported horizontal span like this. That's been known to happen with older real machines, and I suspect it might be even more likely in a virtual cab, because we tend to install more things on the floor.

Customizing cutouts to accommodate the PC

Before finalizing your floor cutouts, you might want to figure out where you're going to place the PC components, so that you can customize the cutouts to better suit the PC. Some particular things to consider:

- If you're going to install the PC in a full case (such as a desktop case or a mid-tower case), you might need to move the subwoofer opening further back to make room.
- The PC needs good air flow for cooling. The air intake openings should be positioned so that they're close to the PC, and so that they'll be unobstructed. If you're installing the PC in a full case, you should figure out where the case's air intake will end up, and place a floor opening at the same spot, so that the case can draw in outside air directly.

See Chapter 27, *Installing the PC* for more on planning the placement of the PC components.

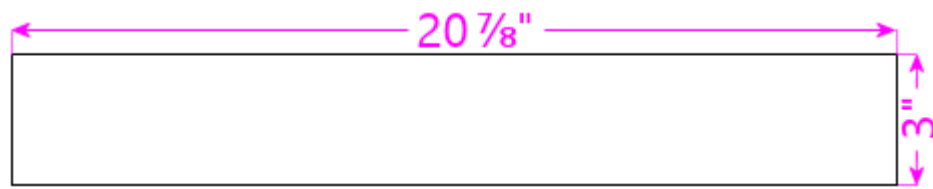
Other floor cutouts

Here are some ideas for other cutouts you might want to make in the cab floor, as long as you're working on this piece. These aren't things you'll find in the real machines, and there's no particular standard place to put them in a virtual cab, but you can consider making provisions for them if they look useful for your build.

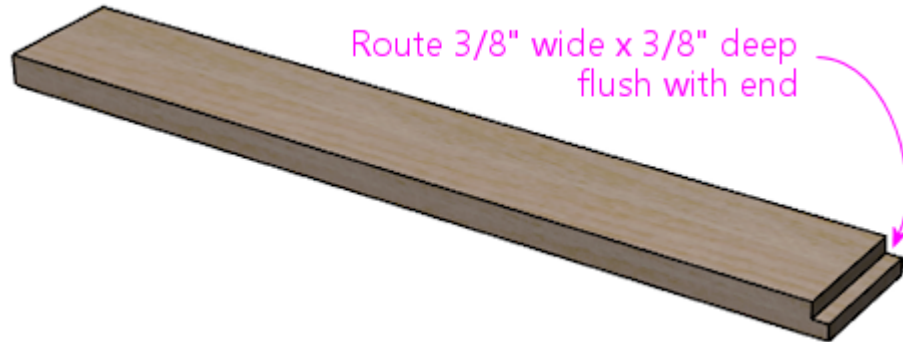
- Keyboard and mouse ports (typically USB). The floor is a good place for ports for input devices that you might want to connect for doing administrative work on the PC, since it's out of sight but within easy reach. I'd pick an area near a front corner, perhaps opposite the power button. Keystone jacks work well for this. See "External I/O plugs" in Chapter 27, *Installing the PC*.
- Openings for undercab light wiring. If you're going to install light strips on the bottom of the cab for ambient lighting, you'll need a small hole somewhere in the floor for the wiring. A 1" diameter hole somewhere along one of the edges is pretty flexible for this purpose. See Chapter 58, *Undercab Lighting*.
- Volume buttons or knob. Some people like to put dedicated volume controls somewhere on the cab, and the bottom (somewhere near the front) is a popular choice because it's out of sight but easily reachable. You can use a volume dial here if your amplifier uses one, or an up/down rocker switch. I installed a rocker switch for this purpose on my own cab, wired through the keyboard encoder to send the Volume Up and Volume Down keyboard commands to Windows. On a new build, I'd probably dispense with the extra switch, and use "shifted" flipper buttons in combination with PinVol.
- Other hidden controls, such as an audio mute button or a "night mode" switch (to silence noisier devices for late night use).

The bottom of the cabinet is a good place for controls that you want to keep hidden but accessible. There's an even better place for controls that you want to be *restricted*, not merely hidden: inside the coin door. Controls located there will not only be out of sight during normal play, but won't even be accessible to ordinary users who don't have the key, preventing kids or guests from messing with anything you don't want messed with. It's the same reason the real machines locate the operator menu buttons on the inside of the coin door. See Chapter 40, *Coin Door*.

Cashbox fence



Cashbox fence. Cut from 3/4" plywood. This piece attaches to the cabinet floor just behind the cashbox area, to form a well for the cashbox. Note: adjust the width for your cabinet width if building a widebody or custom width. The piece should be 3/8" wider than your cabinet's inside width.



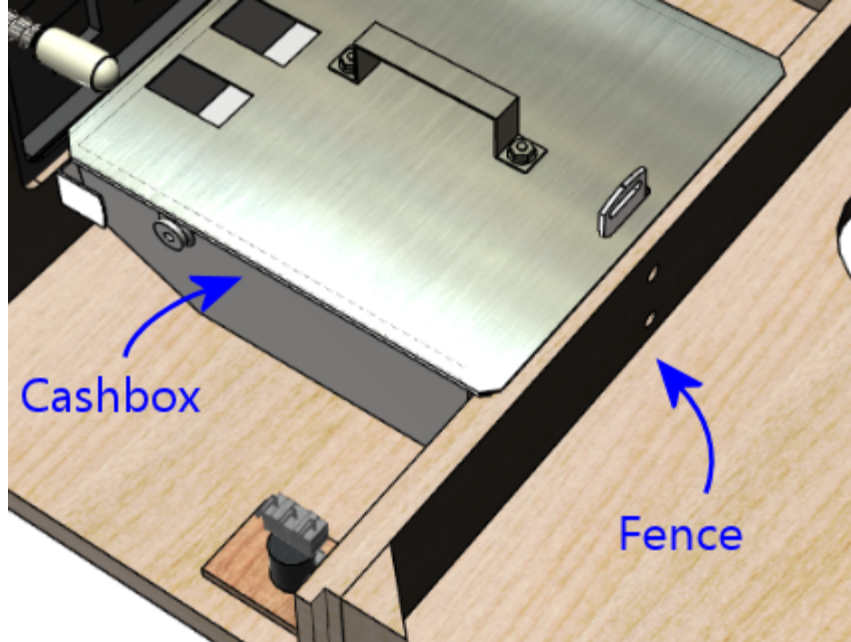
Route a 3/8" wide by 3/8" deep groove, flush with one end. This forms a locking tab that fits into the corresponding slot in the right side wall, for easier assembly.

Note: If you chose not to pre-route the cashbox fence slot in the right side wall, omit the locking tab. Simply deduct 3/8" from the piece's width, and don't route the groove at the end.

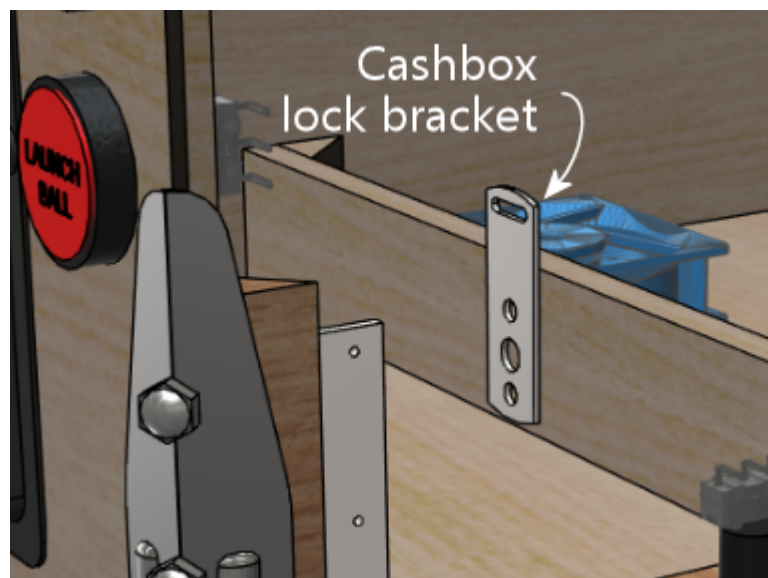
This is a short wall on the floor of the cabinet that delineates the space at the front of the machine where the cashbox goes. On the real machines, it makes a little cubby hole for the cash box and keeps it from sliding around, so that it stays positioned properly under the coin mechanisms.

I don't think most virtual cab builders bother to include a standard cashbox, since it takes up a lot of space at the front. A virtual cab often needs this space for the PC components and other electronics. You'll probably want to skip the fence if you don't use a cashbox. If you do use the standard type of cashbox, though, the fence is worth including, since it holds the box in place.



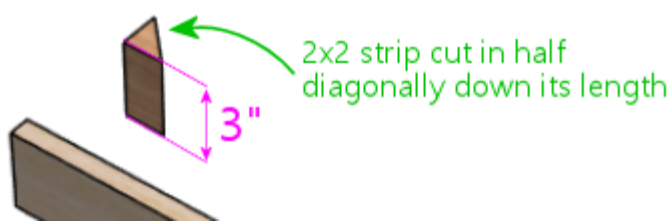


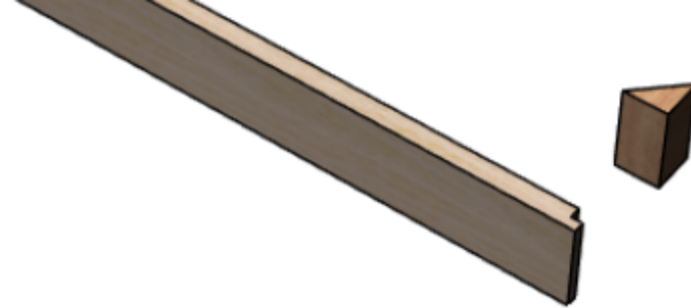
Cashbox fence. The metal tab sticking up is the "cashbox lock bracket", which is attached to the fence, and fits through a slot in the cashbox lid. This is designed to hold a padlock for higher security.



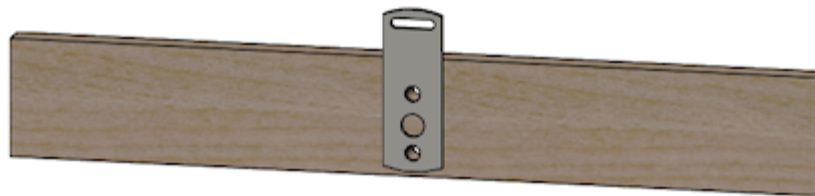
Cashbox lock bracket (Williams/Bally part 01-10030 or 1A-3493-1).

While you're cutting the piece for the fence, also cut two triangular pieces that we'll use to attach the fence to the side walls on the cabinet. These are normally triangular pieces 3" long, which can be made by slicing a 2x2 board in half down its length at a 45° angle. The triangular profile of these wedges isn't important; they just sit in the corners of the fence to help support it. A plain rectangular piece of 1x2 or 2x2 board would be just as functional. If you want to use the traditional wedge-shaped pieces, see Appendix 14, How to Make Corner Braces (and other wood prism shapes) for ideas on fabricating them.





The original machines have a metal bracket in the middle, called the cashbox lock bracket (part number 01-10030 or 1A-3493-1), which can be used with a padlock to secure the cashbox. You probably won't feel the need for such strong anti-theft measures on a home-use machine, but if you want to include the bracket anyway for the sake of completeness, attach the bracket to the fence as illustrated below. Do this before installing the fence in the cabinet, since it'll be hard to drill the holes after it's in place.

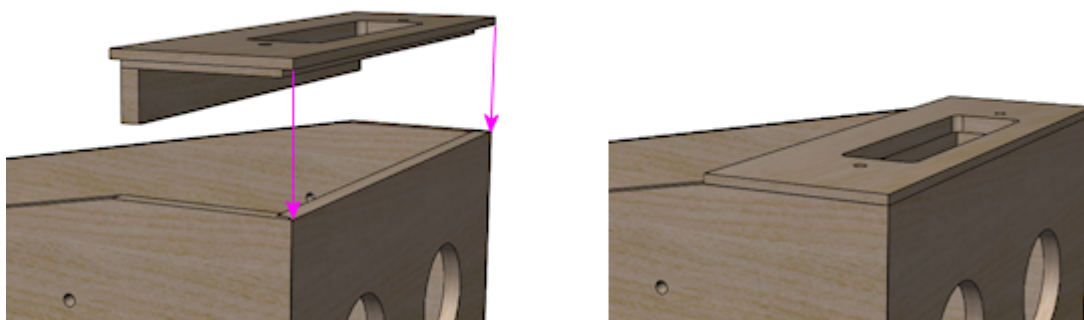


Cashbox lock bracket. Center the bracket left to right, and make it flush with the bottom of the fence. Attach with two #8 x 7/8" machine screws mated with #8 T-nuts, or with #8 wood screws.

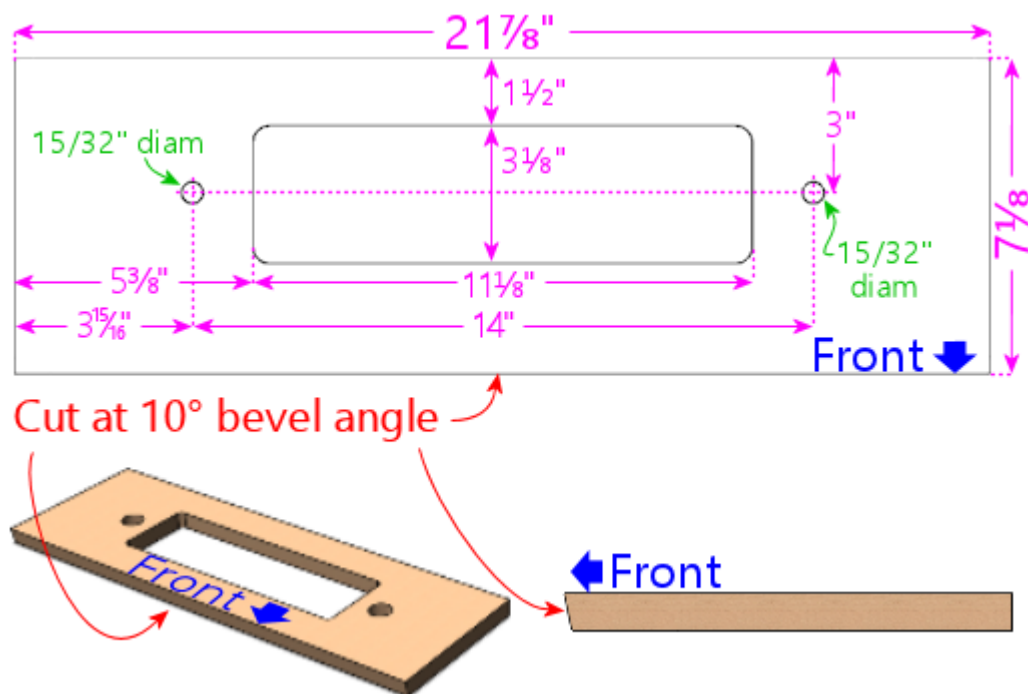
The original WPC machines use two #8 x 7/8" machine screws and #8 T-nuts to secure the bracket; drill 7/32" holes for this setup, using the bracket as a template for the drill locations, and pound in the T-nuts on the back side of the fence. If you want to keep things simpler, use #6 or #8 wood screws instead of the machine screws and T-nuts. That won't be as strong, but it doesn't have to be built like a bank vault for home use.

Rear shelf

This piece sits on top of the cabinet at the very back, where the slope of the side walls flattens to form a shelf for the backbox. The backbox sits on top of this piece. It has openings for the wiring between the backbox and the main cabinet, and holes for a pair of bolts that secure the backbox, so that it can't tip over.



Cutting plan:

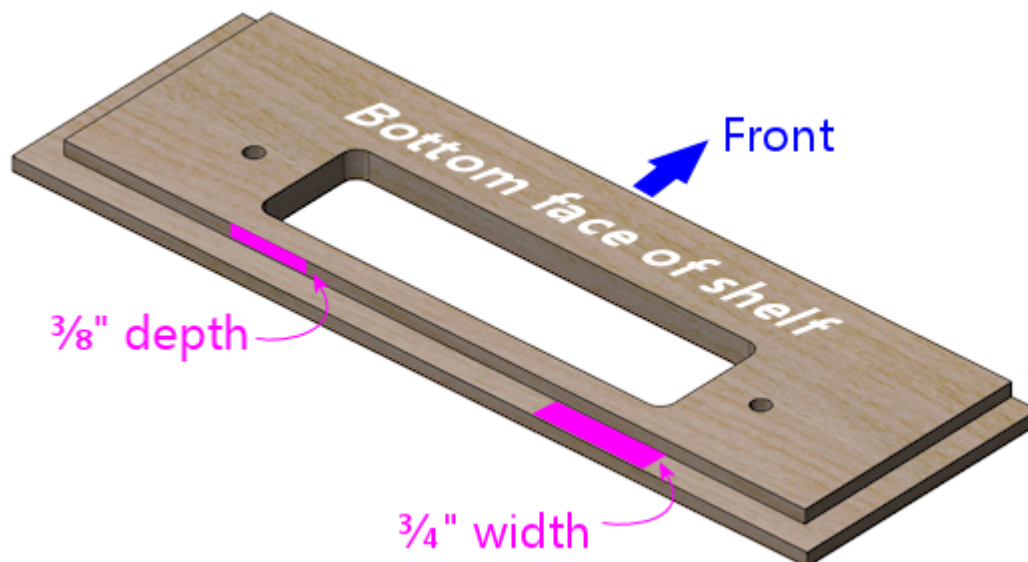


*Rear shelf. Match the shelf's width to the **outside** width of your cabinet. Cut the front edge with a 10° bevel angle, to match the slope of the glass cover.*

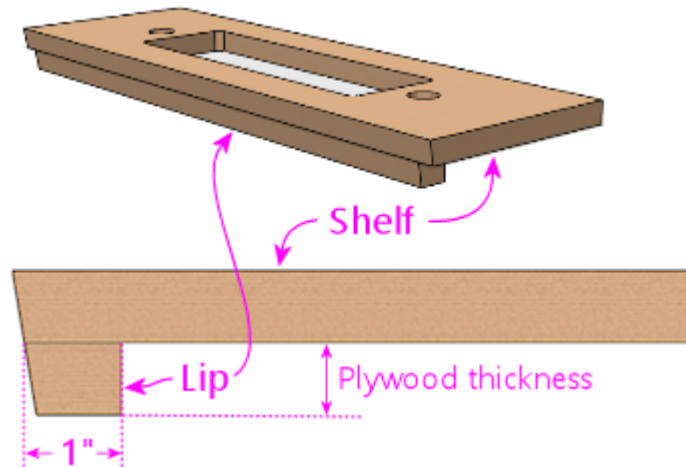
The width shown is for the standard-body WPC cabinet dimensions. If you're using a different cabinet width, match the shelf width to your main cabinet's outside width. Leave the bolt holes and center opening the same size, keeping them centered left-to-right as shown.

Front edge 10° bevel: Cut the front edge with a 10° bevel angle. This matches the slope of the glass cover, so that the plastic trim piece that holds the back of the glass is angled at the same slope as the glass. This makes the glass fit more easily into the trim. If your saw can't make the 10° angled cut, you can cut it square, but it'll make it a little harder to get the glass to fit into the trim.

Routing: Route the bottom edges of the shelf as shown below. This forms a "rabbet" that fits into the top of the cabinet.

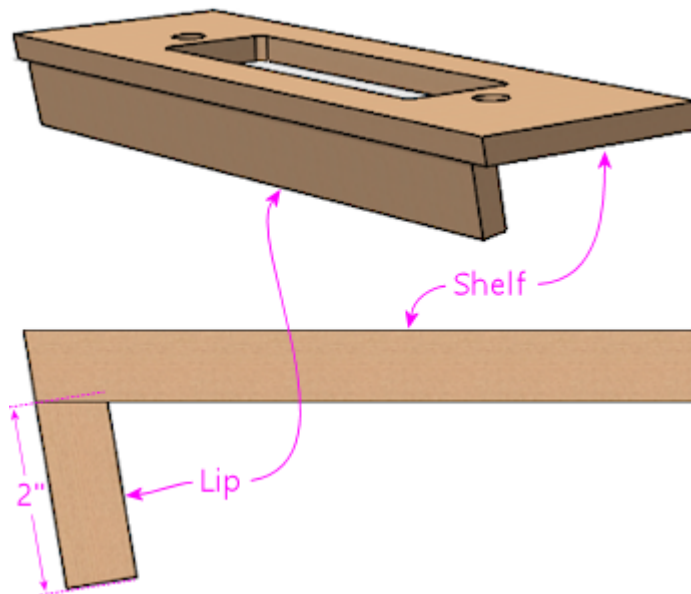


Shelf lip: A separate piece of plywood attaches under the shelf, at the very front edge, to form a "lip". This is mostly to make a big enough area to attach the plastic trim piece that holds the glass. On the original WPC machines, the lip is only as thick as the plywood:



*Shelf lip on the original WPC machines. The lip is simply another piece of plywood, about 1" deep. This should be the same width as the **inside** of your cabinet.*

Optionally, you might want to make the lip extend further down, perhaps 2", like this:



Alternative shelf lip design that extends further down, to cover a larger gap between the shelf and the TV and devices at the back of the machine (e.g., flasher panel).

The height is up to you. It's just a matter of how much space you want to fill between the shelf and the top of the TV and any other devices at the back of the TV, such as a flasher panel. On the original mechanical machines, the lip didn't have to be very tall, because they wanted to leave a lot of space open to make room for tall playfield features at the back, such as ramps and decorations. A virtual cab's "playfield" is just a flat TV panel, though. That might leave a big vertical gap at the back, which you might want to cover with the shelf lip. A lot of cabs use some of that space for flasher panels or LED light strips, in which case the lip can be shorter. If you haven't decided yet how you're going to use the space at the back of the TV, I'd

probably use a short lip like on the WPC machines, and fill any leftover space in the final build with a back wall attached to the TV.

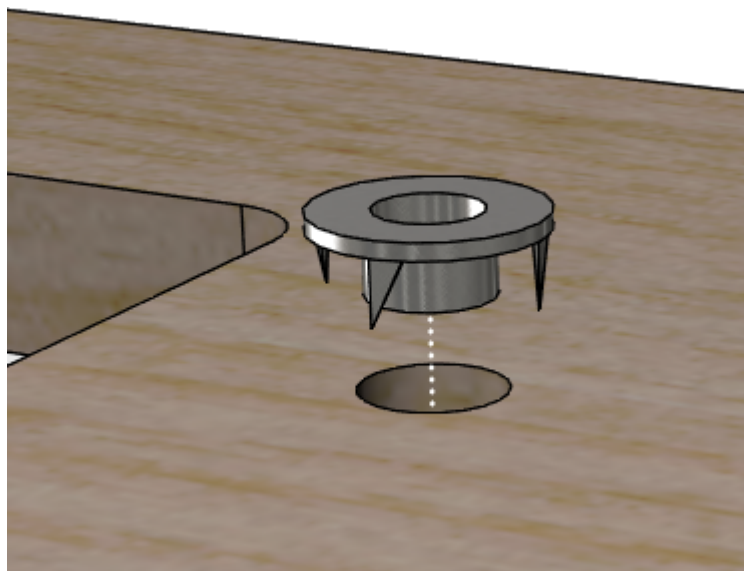
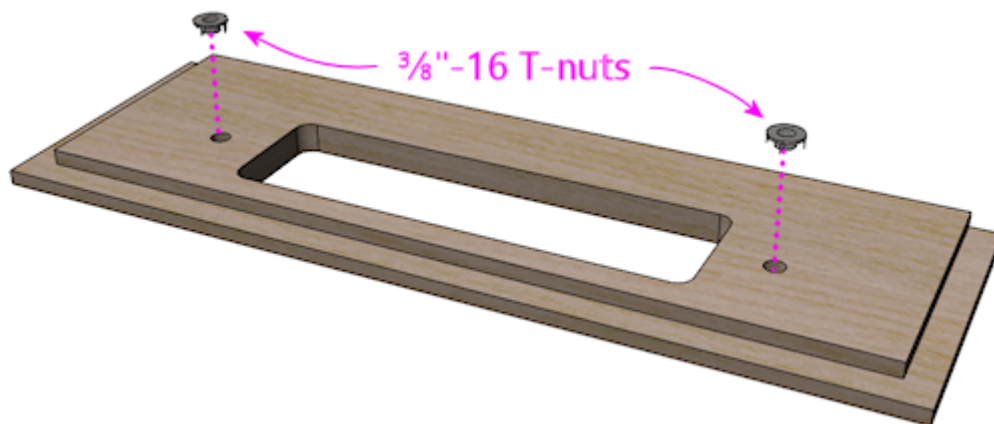
Note that the plywood-thickness lip is probably the smallest you should make it, so that there's a big enough area to attach the plastic trim channel for the back of the glass.

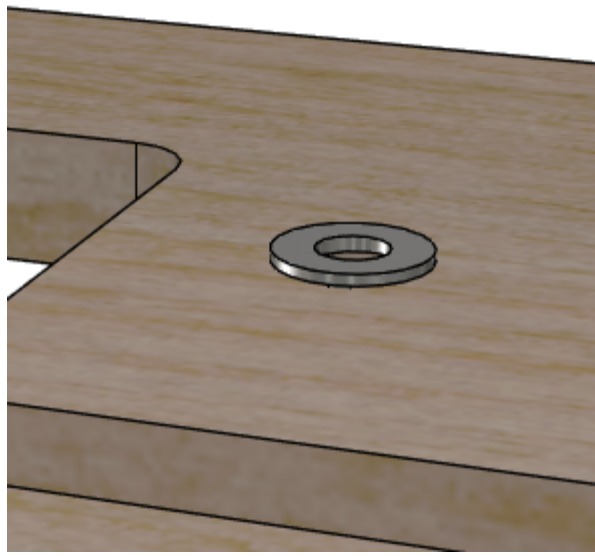
In any case, you should cut the lip with the same 10° bevel angle that you used on the front edge of the shelf so that the slope is continuous across both pieces.

Central wire opening: The rectangular center opening is for routing wire bundles between the main cabinet and the backbox. You can leave this the same size even if you're using a custom cabinet width - just keep it centered left-to-right.

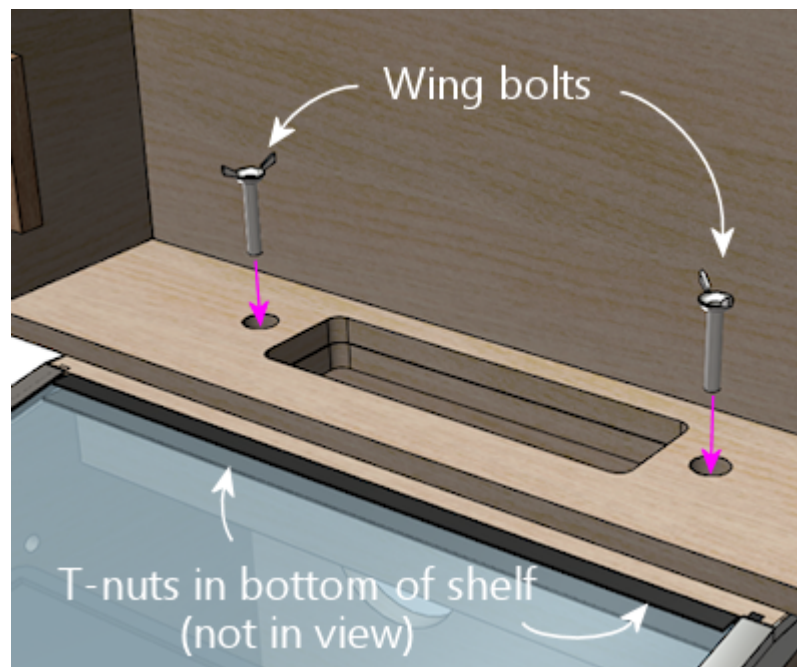
The opening is large enough to accommodate a lot of wiring, but you might need to expand it if you plan to recess an oversized backbox monitor into the main cabinet. If you customize the cutout shape, remember to make the same changes in the cutouts in the backbox floor. For alignment, use the back edge as the reference point, because the backbox's back wall will be flush with the cabinet's back wall when the backbox is installed and placed upright. For left-to-right alignment, use the center point as the reference; the backbox is wider than the shelf, but it'll be centered horizontally when installed, so the center points will line up.

Bolt holes: The 15/32" drill holes on either side of the central opening are for bolts that secure the backbox in the upright position. Install a 3/8"-16 T-nut on the bottom side of each bolt hole. (A T-nut is a type of nut that attaches permanently to a piece of wood.) Insert the barrel of the T-nut into the hole as shown and pound it in to secure it.





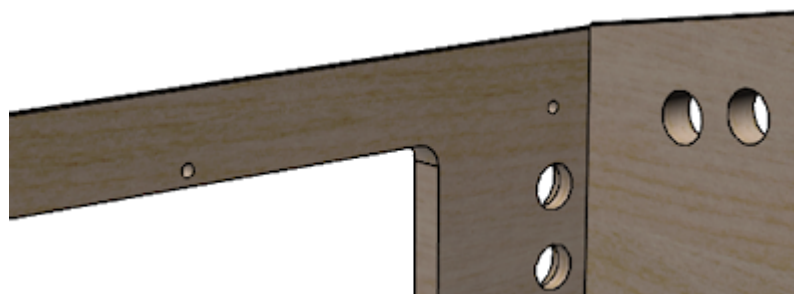
When everything's assembled, these T-nuts will mate with wing screws that you install in the backbox to prevent the backbox from tipping over:

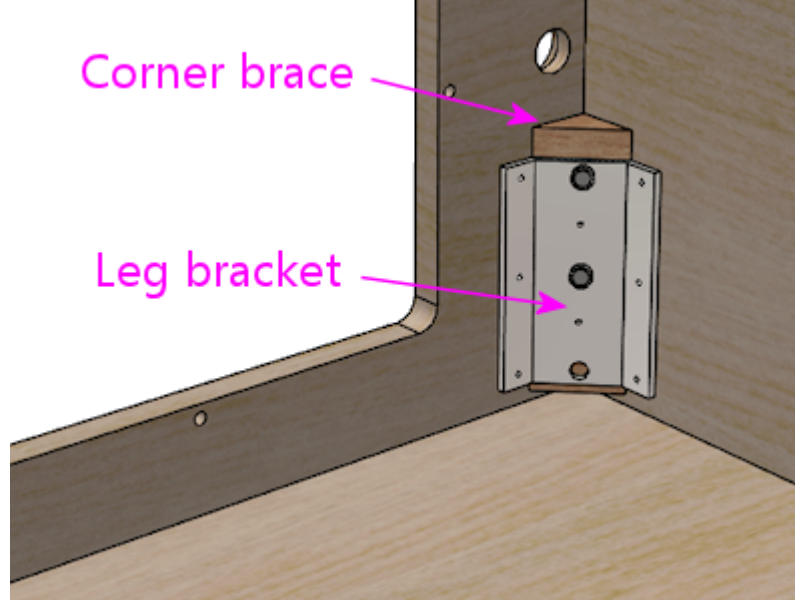


The wing screws are important for safety, so be sure to lay the groundwork for them by installing the T-nuts. Even if you're installing a latch on the back of the backbox, you should still use the wing screws and T-nuts, since most latches aren't strong enough to truly secure the backbox.

Corner braces for the leg bolts

One last detail. We need some corner braces that go under the brackets used to fasten the legs.





The corner braces have a triangular profile, with these dimensions:



The normal way to make these is by cutting a 2x2 roughly in half, down its length, at a 45° angle. This is a bit tricky to do; for help, see Appendix 14, How to Make Corner Braces (and other wood prism shapes).

You'll need four pieces in this shape:

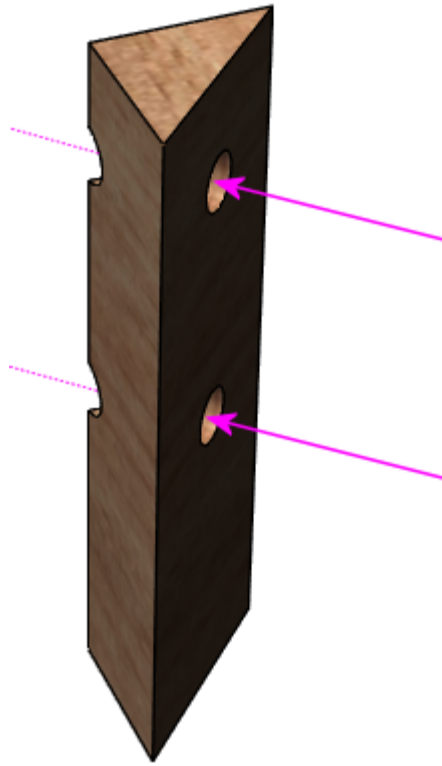
- Two for the front corners, 6" to 8½" long
- Two for the rear corners, 6" to 21½" long

The minimum length is 6" all around, to fill the space under the leg brackets. But you can make them longer if you want, to provide more reinforcement at the corners. At the front, they can reach from the floor of the cabinet to the top of the brackets, which is about 8½". They shouldn't go any higher than that, because if they did, they could get in the way of the plunger and front panel buttons. At the back, they can reach from the floor to the top of the cab, which is about 21½".

If you wish, you can pre-drill the holes for the leg bolts. (This is best if you're using the minimum 6" length. If you're making the braces taller, you'll probably want to install them in the corners first, and then drill holes from the outside, to make sure they line up with the holes in the cabinet corners.)

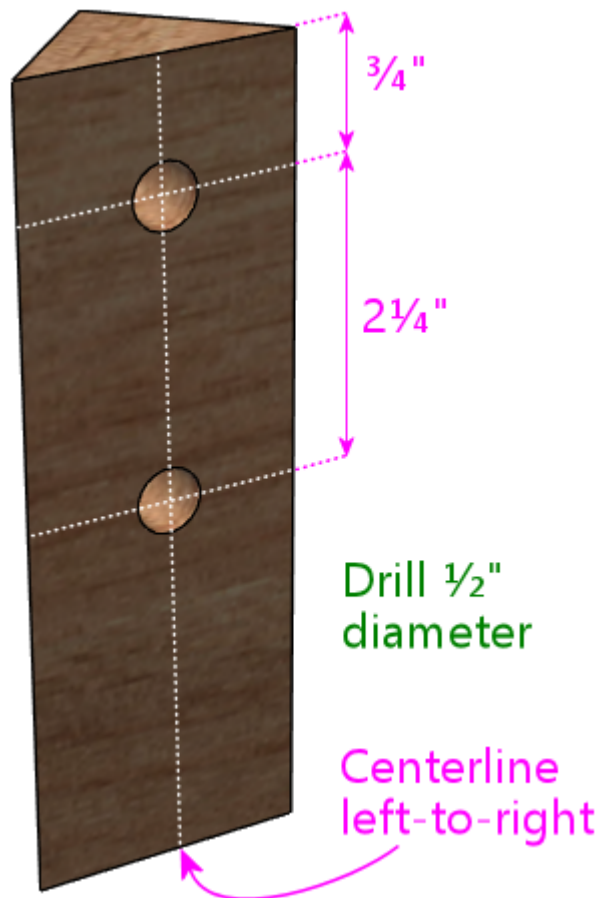
If you want to pre-drill the holes, drill ½" diameter holes in each piece. You can also just use your brackets as drilling templates. Drill through the diagonal face (the

widest face), on the centerline, square through that face towards the opposite corner.



Drill starting on the diagonal (widest) side, square into that face. The hole should come out on through the opposite corner.

One hole will be near the center (vertically) of the wood piece, and the other will be near the edge. Make four identical copies of this piece.



Backbox

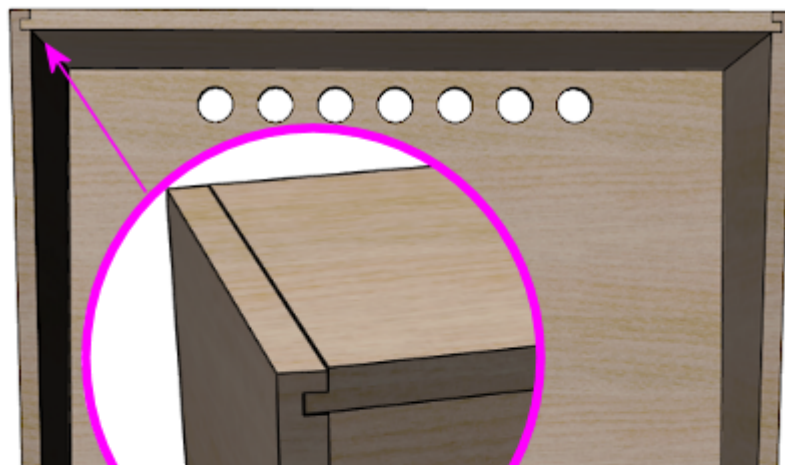
The backbox is (happily) a whole lot simpler than the main cabinet. It doesn't have as many cutouts, and we don't have to get as fancy with the corner joins. The top and bottom surfaces are typically out of view, so we can use joins that leave seams, by hiding the seams on the top and bottom edges where they won't be seen. All of the joins for the backbox can be accomplished with straight router bits.

The backbox is *mostly* built from the same $\frac{3}{4}$ " plywood used in the main cabinet. There's one exception, though: the back wall is made from $\frac{1}{2}$ " plywood. The original WPC backboxes had $\frac{1}{2}$ " thick back walls, so we're sticking to the same plan to keep the interior dimensions the same.

If you want to substitute $\frac{3}{4}$ " plywood for the back wall, it's fairly easy. You just have to adjust the routed grooves in the other walls where the back wall joins to accommodate the thicker panel. We'll give you a reminder about that when we get there.



Exploded view of backbox





Corner joins, viewed from the front. This type of join leaves a seam along one face, but we orient the joins to place the seams along the top and bottom , which are normally out of view.

(For purists, I have to confess that my design has a discrepancy from the original WPC design, which is that the joinery at the bottom of the back wall is different. The original WPC back wall simply butts up against the floor, whereas my design uses a rabbet to match the other three sides of the back wall. This is obviously an inconspicuous area - I didn't even notice the deviation until some time after publishing this chapter - and I can't think of any way the difference affects function. The rabbet is perhaps a bit stronger, and works as a glue joint, whereas the original butt join requires nailing. Williams might have used the original butt join simply for manufacturing convenience: it's more forgiving of measurement errors, in that you don't have to get the bottom and back pieces to line up precisely. If you want to modify my plan to reinstate the original design, cut the back wall $3/8$ " taller, make the floor piece $1/2$ " less deep, and omit the rabbet at the back of the floor.)

Translite and DMD guides

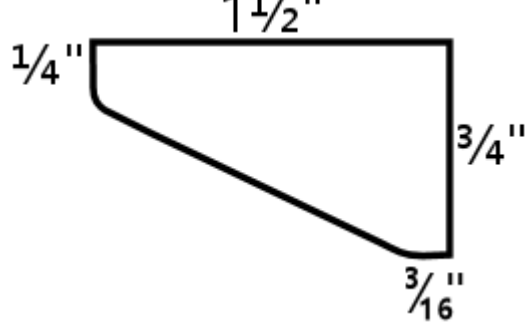
The backbox requires some simple rectangular wood strips that acts as guides to hold the translite and speaker/DMD panels in place.

Quantity	Material	Dimensions
2	$1/2$ " plywood	$4\frac{3}{4}$ " x $3/4$ "
2	$1/2$ " plywood	15" x $3/4$ "
1	$3/4$ " plywood	$27\frac{1}{8}$ " x $3/4$ "
2	$3/4$ " plywood	$12\frac{3}{8}$ " x 1"
1	$3/4$ " reducer molding or nominal 1x2 square-edge board, cut to a similar shape (see diagram below)	$27\frac{1}{8}$ " length

The plywood pieces aren't visible to players, so don't worry about making the edges look pretty. They're all hidden behind the backglass or speaker panel when the machine is assembled.

The "reducer molding" shape is a more challenging trim piece that requires an angled cut. And this one *is* visible to players - in fact, its whole purpose is cosmetic - so you'll want to make it look nice.

One way to make the molding piece is to start with a nominal 1x2 board, at least 28" long, and cut it lengthwise ("rip" it) with a diagonal cut, so that it has this approximate cross-section:

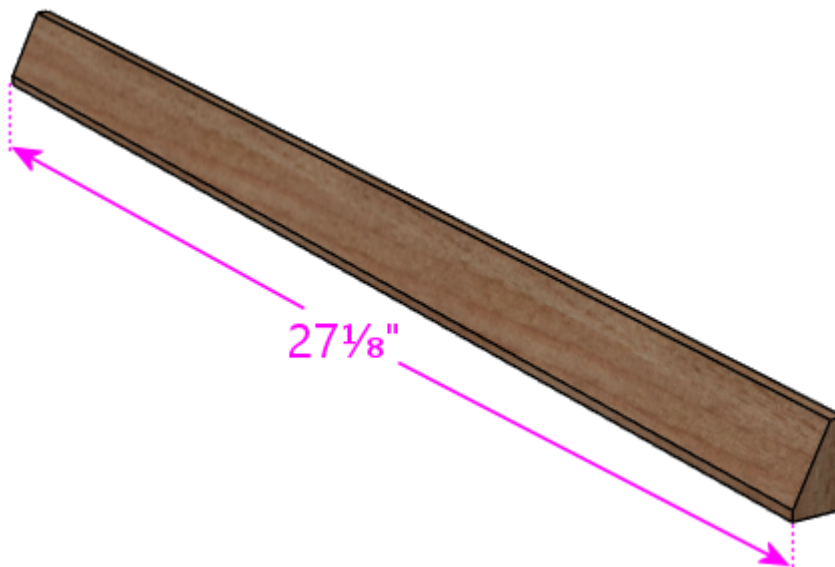


For help cutting this shape, see Appendix 14, How to Make Corner Braces (and other wood prism shapes).

The diagram above is based on the molding used in the original Williams machines, but you don't have to reproduce the shape perfectly, because this piece is purely cosmetic - the shape doesn't have to mesh with any other parts and doesn't serve any mechanical function. It's just there to hide the top glass channel and give it a finished look. The only important thing is to give it a pleasing tapered shape. The rounded corners are likewise not required, but they look a little nicer; you can get that effect simply by sanding the corners until they start to round out.

An alternative to cutting this shape yourself is to buy a pre-cut wood molding in roughly the same shape. There's a common type of floor trim called a **3/4" reducer molding** that has roughly this same shape and size. A 3/4" reducer molding will typically be a bit deeper than the profile we want, and it'll usually have a tab that sticks out from the back. You'll need to cut off the tab if present.

Once you have a strip with the right profile, cut it to a length of $27\frac{1}{8}$ ".

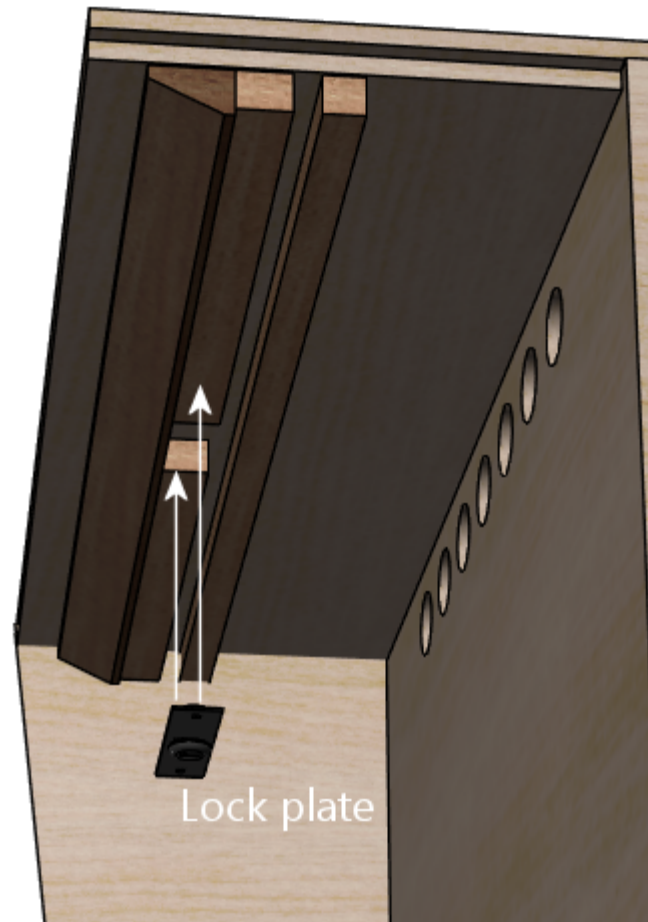


Translite lock plate preparation

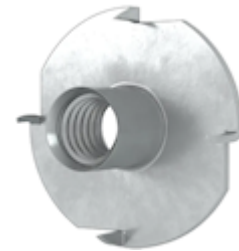
If you're planning to install a translite lock plate, there's some preparation you can do at this stage that will make installing the lock easier and more secure when you get there. If you don't know what the translite lock plate is, you can learn about it in the "Translite lock" section in Chapter 23, Cabinet Hardware Installation. Briefly, it's a keyed lock that you can install at the inside top of the backbox to secure the translite. For a home machine, the security function isn't important, but you might want to include the lock anyway if you're a stickler for realism, since it's there on all of the real machines going back at least to the 1980s.

The translite lock is installed in the front translite guide (described in the section

above), which is part of the ceiling of the backbox. The front guide has a gap of about 2 inches in the middle, specifically for the lock.



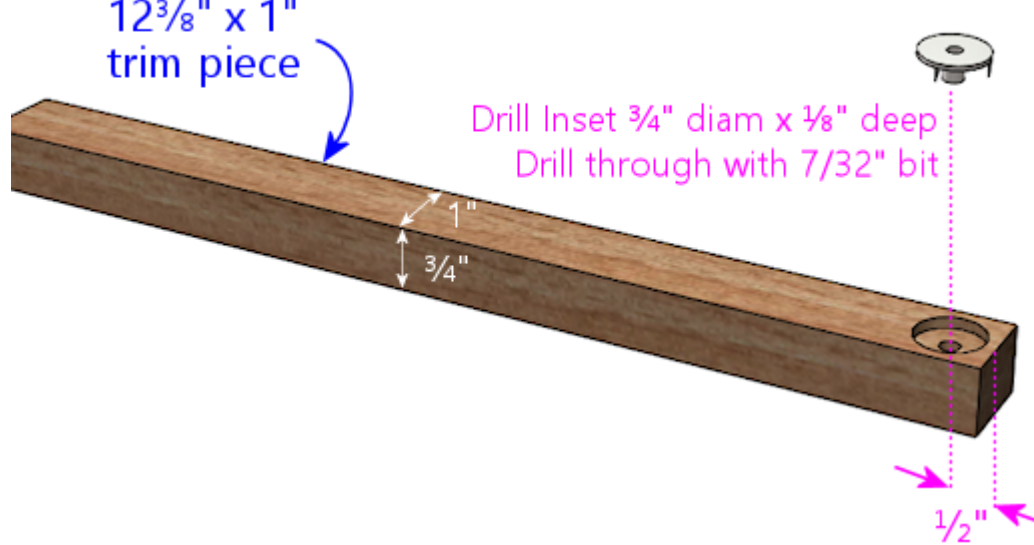
On the real machines, they install the lock plate with #8-32 "security" Torx machine screws, which have a special version of the Torx head that's meant to be tamper-resistant. These are **machine** screws, not wood screws, so they can't self-tap in wood; they need to be fastened with nuts. If you look at the arrangement pictured above, you can see that there's no way to install ordinary hex nuts by hand with this setup, because you'd have to get behind the wood trim somehow - and that's going to be glued in place by the time you're ready to install the screws. So the question is: how do you install a nut in a place you can't reach? The answer is a T-nut. A T-nut is threaded like a hex nut, but it's permanently installed in the wood rather than being screwed on by hand. They're specifically for this type of situation where you need to pre-install a nut someplace that'll be inaccessible after assembly.



So, **if** you want to install the lock the original way, the required preparation is to pre-install T-nuts in the 12 $\frac{3}{8}$ " x 1" trim pieces:

- Mark the drill center at 1/2" from one end, centered across the width
- Using a 3/4" Forstner bit, drill an inset on the marked center about 1/8" deep, to countersink the T-nut
- Drill the rest of the way through on the same center with a 7/32" bit
- Insert a #8-32 x 1/4" T-nut from the recess side
- Pound it in flush into the recess

#8-32 x 1/4" T-nut

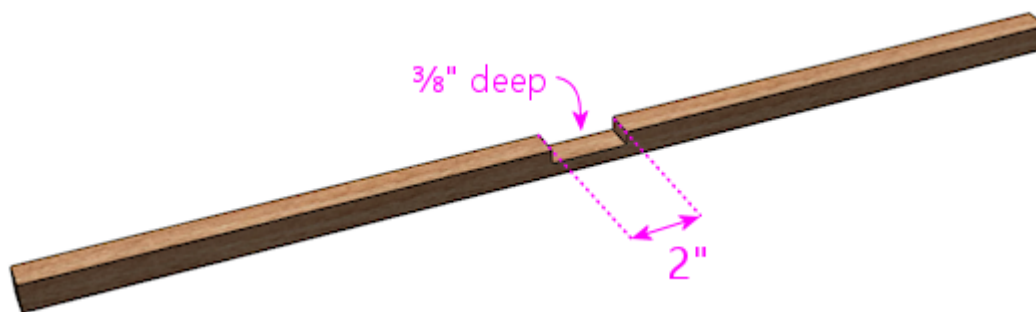


Simpler alternative: rather than pre-drilling and installing the T-nuts, you can discard the Torx screws that come with the lock plate kit, and attach the lock to the trim with ordinary wood screws instead. That eliminates the need for the T-nuts. The reason they use the machine screws and T-nuts on the commercial machines is to harden the lock against prying. You're probably not as concerned with that in a home-use machine, in which case wood screws are fine. You can install wood screws after the trim is installed, so there's no need for all this prep work if that's what you're going to use.

Extra routing for translite lock

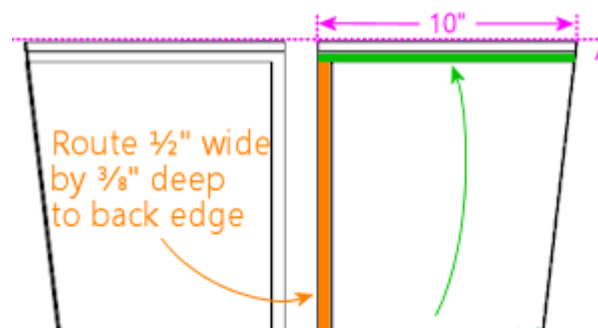
There's a little extra routing you need to do to ensure a good fit for the translite lock. You can skip this if you're not installing a lock.

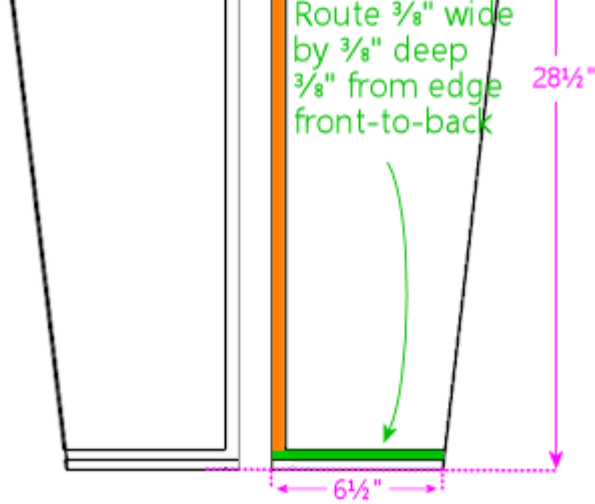
In the $27\frac{1}{8}" \times \frac{3}{4}" \times \frac{3}{4}"$ piece, route a 2" wide notch in the center of one side, to $\frac{3}{8}"$ depth.



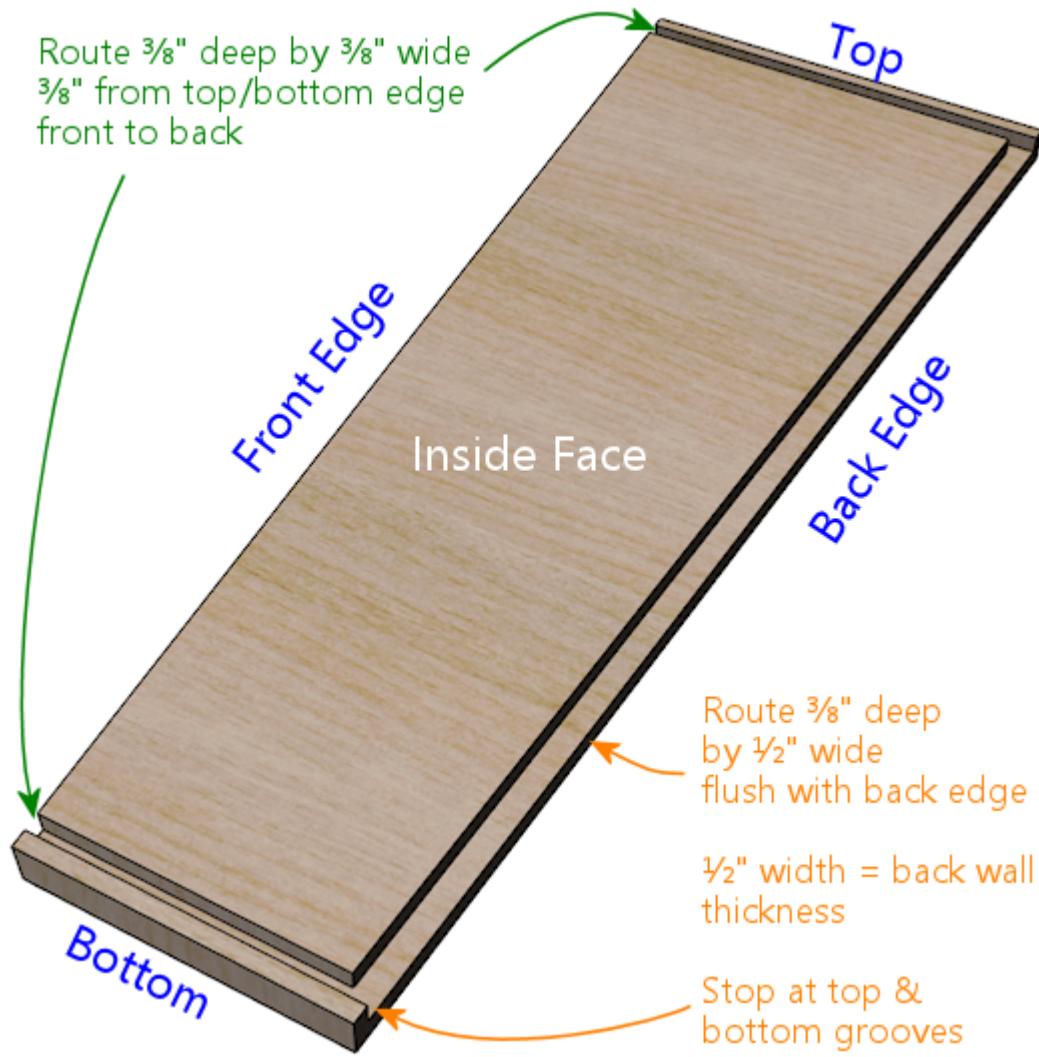
This is necessary to leave room for the lock tab when it's in the "locked" position. The tab is slightly wider than the slot, so it needs this extra room on the other side.

Backbox sides





Backbox left and right sides, shown from the interior side to detail the routed grooves for the joins. These are mirror images of one another. Note that the rear groove's width should equal the thickness of your back wall plywood. Our plans assume you're using $\frac{1}{2}$ " plywood for the back wall, so the groove is shown at $\frac{1}{2}$ " width.



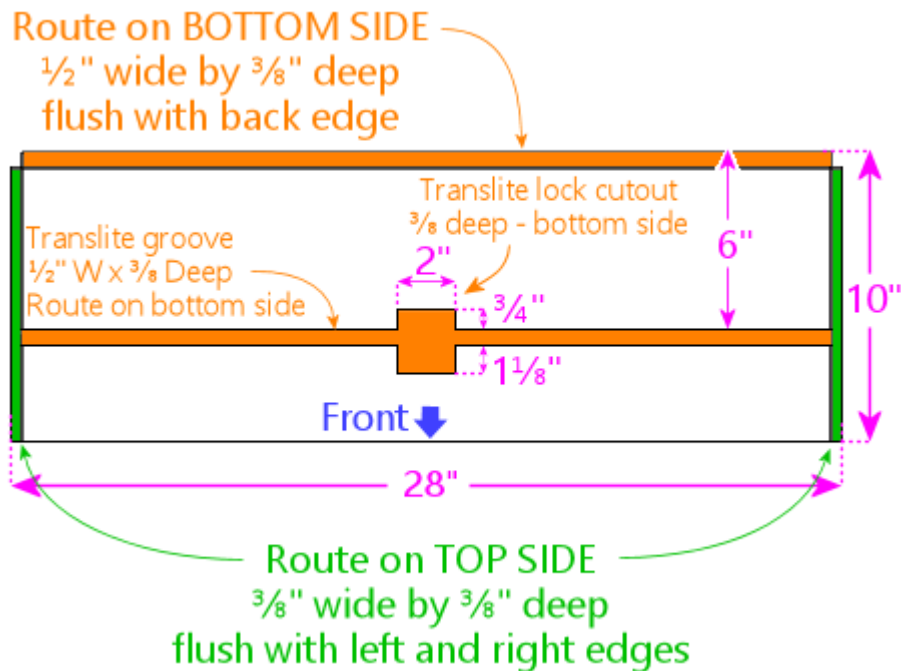
3D view of the routed grooves in the side walls, to clarify the geometry.

The routing at the back edge assumes you're using $\frac{1}{2}$ " plywood for the back wall. If you're using a different thickness, simply increase the width of the groove to match the thickness of your back wall.

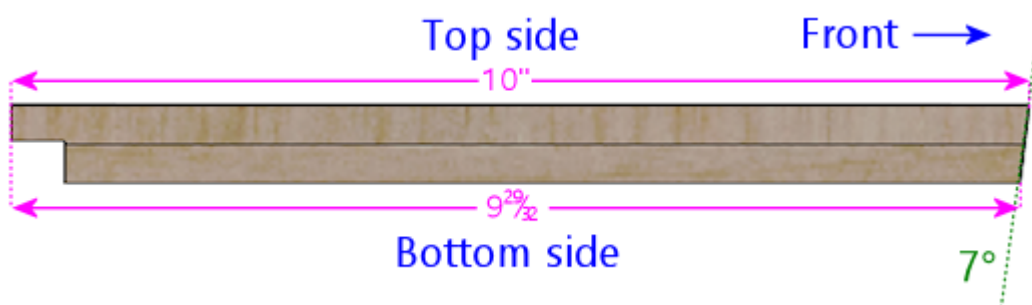
Backbox top

The top of the backbox has a few special features:

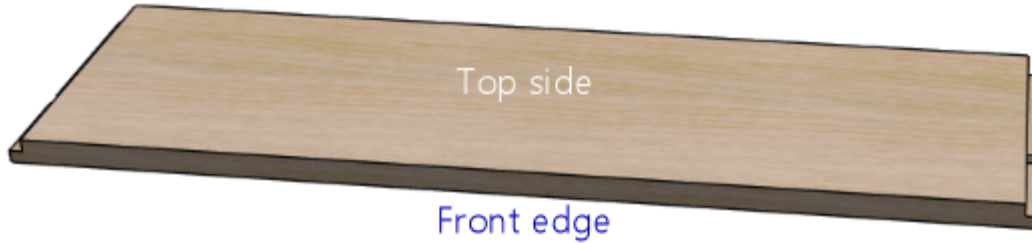
- The front edge should be cut at a 7° angle to match the slope of the front edges of the side walls.
- The side edges are routed on the top side in a rabbet cut, to fit the rabbet grooves in the side walls.
- The back edge is routed on the bottom side in another rabbet cut, to fit the back wall.
- A 1/2" wide, 3/8" deep groove runs across the width of the bottom side of the piece. This matches the plane where the translite fits. The translite doesn't actually sit in this space most of the time, but this groove provides a little extra room to lift the translite into when inserting and removing it. You can omit this if you're not using a standard translite.
- A 2" wide rectangular depression is routed in the middle of the translite groove, on the bottom side of the piece, to accommodate the translite lock. Center this side to side, and refer to the diagram below for the dimensions. This is only needed to make room for the translite lock, so you can omit it if you're not using a lock.



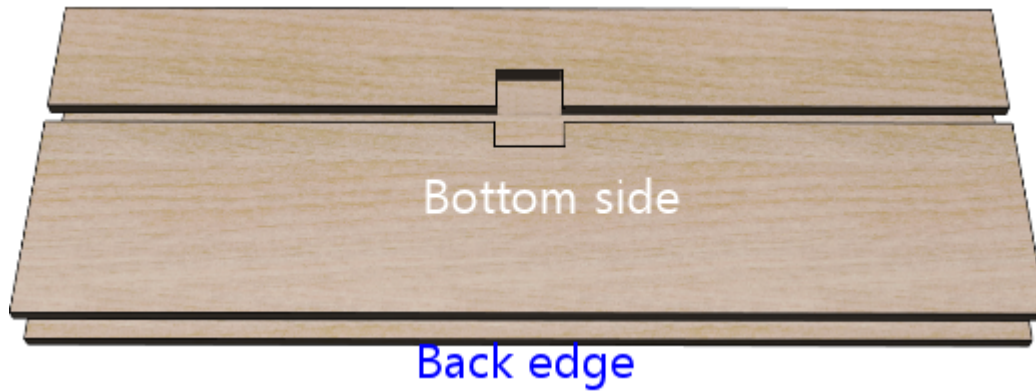
Backbox top piece (roof)



To match the slope of the front sides, the front edge of the top piece should be cut at a 7° angle. This is an edge-on view from the left side.



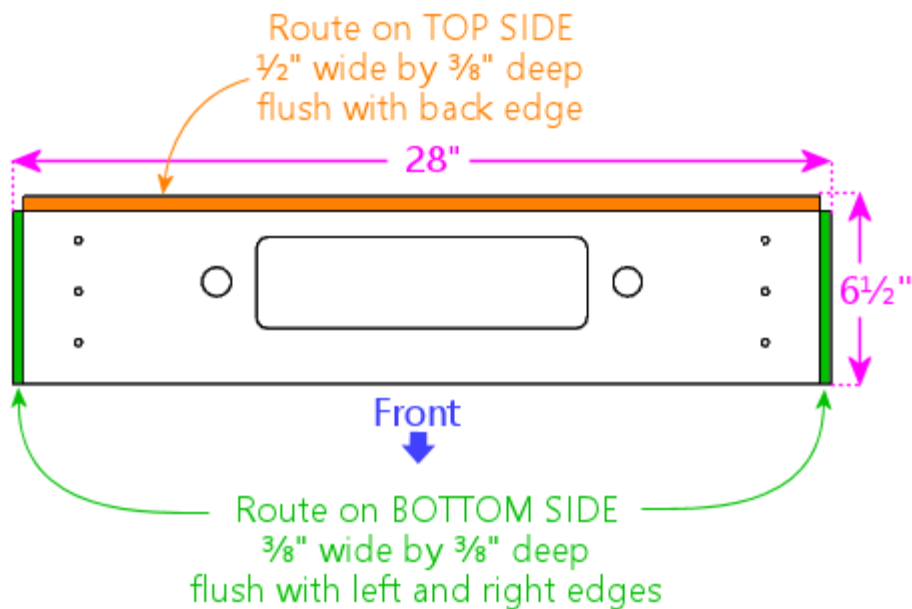
3D view of top piece, viewed from top front, to show routing detail on the top side.
The grooves at the wide are $\frac{3}{8}$ " deep and $\frac{3}{8}$ " wide, all the way to the outer edges.



Top piece, bottom side, viewed from the back, to show routing detail on the bottom side.

If you're planning to install any "toppers" (decorations on top of the backbox, such as a rotating beacon, fan, bell, or flashers; see Chapter 42, Backbox Toppers), consider pre-drilling any openings in the roof that will be needed for mounting hardware or wiring.

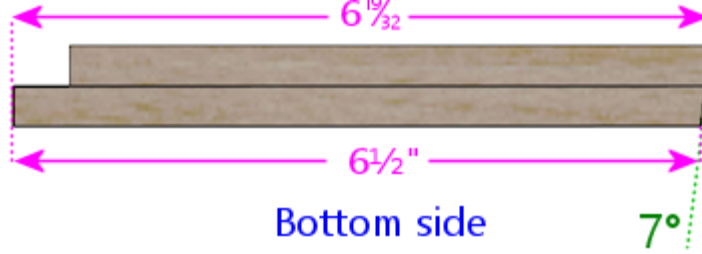
Backbox floor



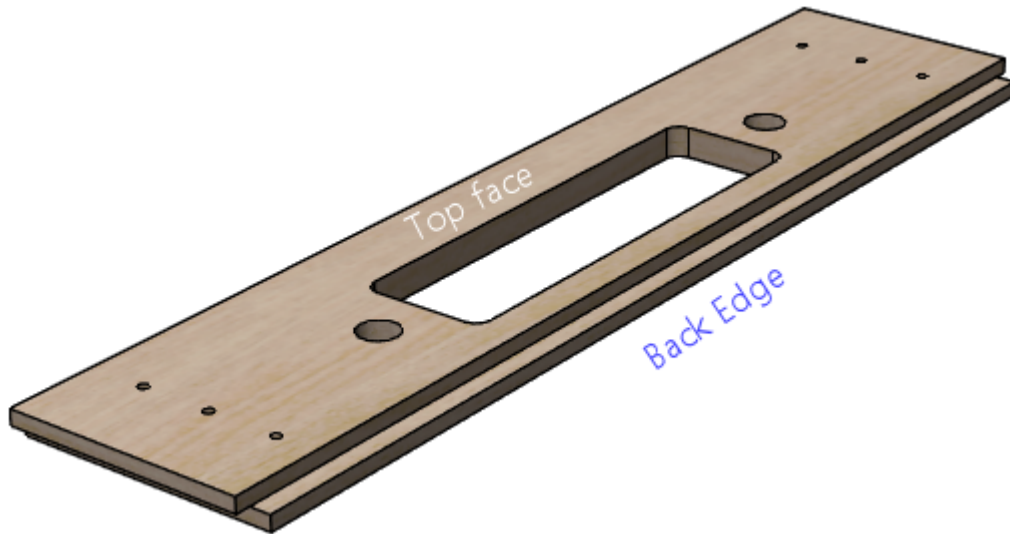
Backbox floor

Top side

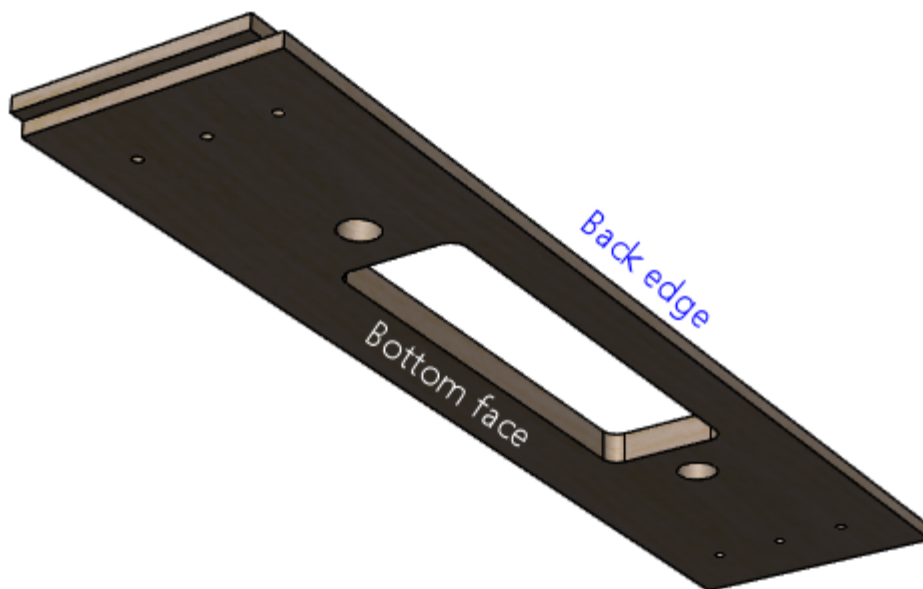
Front →



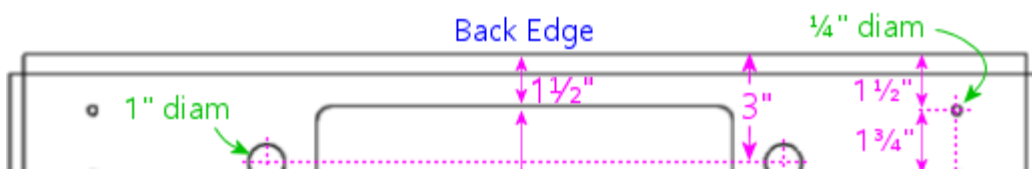
To match the slope of the front walls, cut the front edge at a 7° angle. This is an edge-on view from the left side.

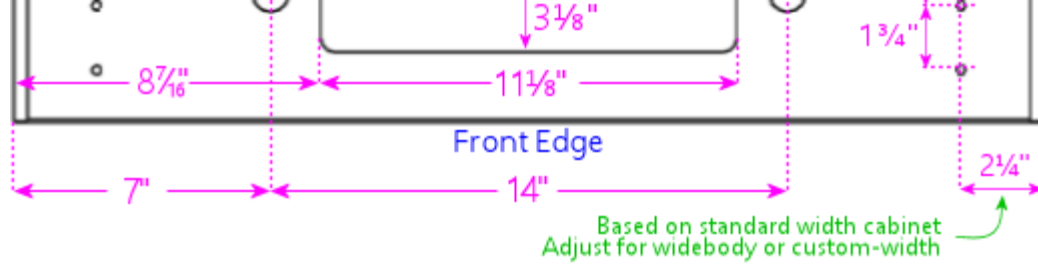


3D view of backbox floor, viewed from back side, to show routing detail. The groove at back is $\frac{3}{8}$ " deep and $\frac{1}{2}$ " wide, flush with the back edge. The $\frac{1}{2}$ " width should match the plywood thickness of the back wall.



Backbox floor, bottom side, to show routing detail.





Above: Cutouts in floor of backbox. The rectangular cutout is for passing cables between the backbox and cabinet. The 1"-diameter holes on either side of the cable cutout are for safety bolts that lock the backbox in the upright position. The 1/4"-diameter holes along the outer edges (three on each side) are for the WPC-style hinge brackets that attach the backbox to the main cabinet. The hinge bolt positions shown are for a standard-width main cabinet - they need to be adjusted for a widebody or custom-width cabinet (see below).

Cable cutout: The rectangular center cutout is meant to match the corresponding cutout in the "shelf" at back of the main cabinet. If you're customizing the shape of the cutout, remember to make the same changes in both places. To figure the alignment between the two parts, use the back edge as the reference point in both places. When the backbox is installed and placed upright, its back wall will be flush with the back wall of the main cabinet. For left-to-right alignment, use the center point as the reference: the backbox is wider than the shelf, but it will be centered side-to-side when installed, so the center points will line up.

Lock bolts: The 1"-diameter holes on either side of the center cutout are for locking bolts. These should be aligned on the same centers as the corresponding 1/2" holes in the main cabinet shelf. If you had to move those to accommodate a custom center cutout, move these holes to match. Note that the shelf holes are 1/2" diameter, whereas the corresponding backbox are 1" diameter. As with the center cutout, the reference point to use for alignment is the centerpoint of the back edge, because that will line up on the shelf and backbox.

Hinge bolts: The 1/4" diameter holes near the outer edges (three on either side) are for carriage bolts that attach the WPC-style hinges to the backbox. Drill these only if you're using the WPC-style hinges.

Important! The positions shown are for a standard-width main cabinet. If you're using a widebody or custom-width cabinet, **or** a custom backbox width, you'll need to refigure the positions. Use this formula:

$$\text{Inset} = (\text{Backbox Width} - \text{Cabinet Width} - 2\frac{3}{8}") \div 2$$

Plug in the **outside** widths of the backbox and cabinet (as they will be when assembled). The result is the inset of the bolt holes from the left and right edges of the floor, so simply substitute this for the measurement shown in our diagram.

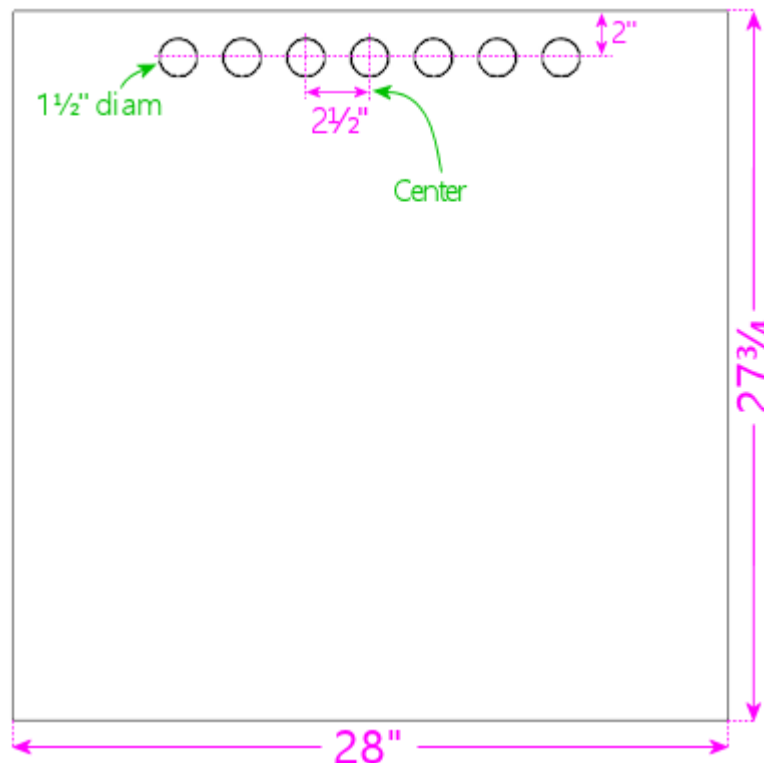
If you don't want to take chances on getting the measurements perfect before-hand, you can wait to drill these holes until you've assembled your cabinet and backbox, at which point you can set it up and use the hinges themselves as a drilling template to mark the proper positions. This is getting a little ahead of ourselves, but here's the procedure:

- Make sure the shelf is in place in the cabinet, if you haven't already installed it. No need to glue it yet; just set it in place.
- Attach the hinges to the main cabinet using their pivot bolts. They'll rotate freely, so be careful not to let them scratch anything.

- Put the backbox in position. Center it left-to-right, and align the back wall of the backbox so that it's flush with the back wall of the main cabinet. The front of the shelf will stick out slightly further than the front of the backbox; that's normal. Have an assistant hold it up so that it doesn't fall over from this precarious position - it's heavy enough to be dangerous!
- Rotate the hinges up into position where they'll attach to the backbox. Make sure the contact area is flush with the bottom of the backbox. Mark bolt hole positions.
- Take down the backbox and drill at the marked positions.

Backbox back wall

The back wall of the backbox on the real machines is typically 1/2" plywood (or particle board or MDF, if you prefer). It's a simple rectangular piece, with some holes for passive cooling.



Backbox back wall. The holes near the top are for passive ventilation. Note that the back wall uses 1/2" plywood.

Backbox ventilation

The original WPC backboxes used passive ventilation, via seven 1 1/2"-diameter holes along the top of the back wall. ("Passive" meaning that they didn't use fans to circulate air; they relied on natural air flow driven by hot air expanding and rising.)

Some virtual cab builders add powered fans to the backbox for extra cooling. My experience has been that this passive cooling is adequate for the backbox (see Chapter 28, Cooling Fans), but some people are concerned about heat from the backbox video displays. If you want to add active cooling, I'd remove the passive vent holes and replace them with one or two larger circular openings for 120mm PC case fans, similarly placed near the top of the back wall. You could also add some intake vents at the bottom, although I don't think that's necessary, as air will be drawn in from the main cabinet through the openings in the backbox floor.

If you add cooling fans, be aware of the space requirements for the other equipment

plan to install in the backbox, such as the TV, DMD, replay knocker, and bells.

Backbox back door

Some virtual cab builders make the back of the backbox into a door rather than a fixed wall.

The plan I'm presenting here uses a fixed back wall, following the original Williams design. On the real machines, most of the main control electronics are mounted on this wall - the CPU board, sound board, power supply board, etc. To access these parts for service, the operator simply removes the translite and accesses the interior from the front side.

The complication for a virtual cab is that we fill most of the backbox with a TV. Some cab builders mount the TV in such a way that it can't be easily removed, in which case you won't be able to access anything behind the TV through the translite side. That's where a back door comes in handy.

I don't have an alternative set of plans to offer using the back door approach, so if you want to go that route, you'll have to improvise something. Other people have built such schemes into their cabs, so you might be able to find ideas by checking build threads on the forums.

I personally prefer the fixed back wall, instead of a door. The main reason is that it makes the backbox a lot stronger if the back is a solid, fixed panel. I also don't like the idea of using a permanent mounting for any of the TVs, since doing so makes it very difficult to repair or upgrade the machine later. I prefer to install all of the TVs (and other major components) in such a way that you can remove them non-destructively when necessary. In the case of the backbox TV, I favor mounting it so that it can be removed through the front of the backbox, preferably without having to disassemble anything. That removes any need for a back door. It also makes it easy to replace the TV, if that should ever become desirable or necessary.

See Chapter 30, Backbox TV Mounting for some ideas about how to mount the TV so that it can be removed.

Toppers

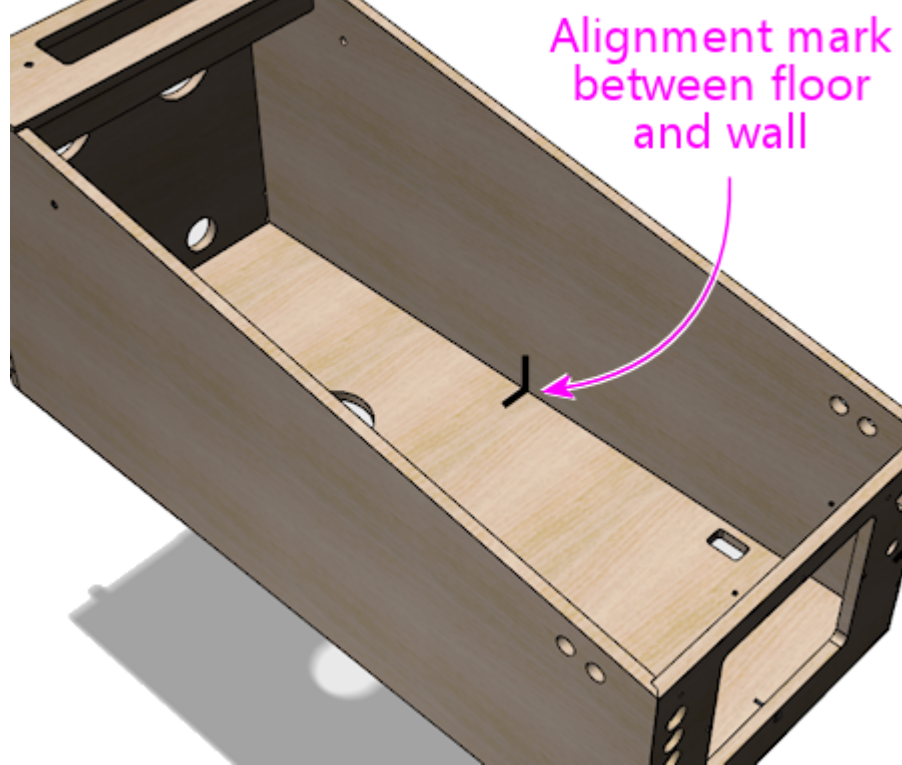
If you're installing some kind of "topper" (a decoration on top of the backbox, such as a beacon, fan, bell, or flashers: see Chapter 42, Backbox Toppers), consider pre-drilling any openings needed for the mounting hardware and wiring.

How to assemble the cabinet

Before you glue everything together more or less irrevocably, it's a good sanity check to do a "dry fit" of the pieces (fitting them together without any glue or nails) to check that everything is the right size and aligns as expected. Check for any dados that are too tight, and use sandpaper or a file to expand them slightly as needed. Check the alignment of the rabbet joints.

Tip: When you do the dry fit, once you have the whole thing together and you're happy with the fit of all of the corners, **make alignment marks** showing the current alignment of the floor with all four walls. In other words, draw a little line at the inside seam between the floor and the wall, somewhere along the seam. When you're doing the final glue assembly, you can use the marks to re-align the floor at the same, known-to-work position. This is helpful because it can be difficult to slide the floor around in the groove once the glue is applied, so it's good to be able to get it in exactly the right position on the first try.





Have a good quality wood glue on hand. This will be used at all of the joints. Optionally, you can also use finish nails (perhaps $\frac{3}{4}$ " #18 brads) along the seams, spaced a few inches apart. Nails will add some strength and will serve to hold the joints in place while the glue dries, but the trade-off is that they create a certain amount of risk of splitting wood around the edges. I used nails for my own build, but I don't think they're really necessary. If you're using the joints we suggested (dadoes at the floor seams, and either the mitered rabbet or double rabbet joints at the corners), I think glue alone will be plenty strong.

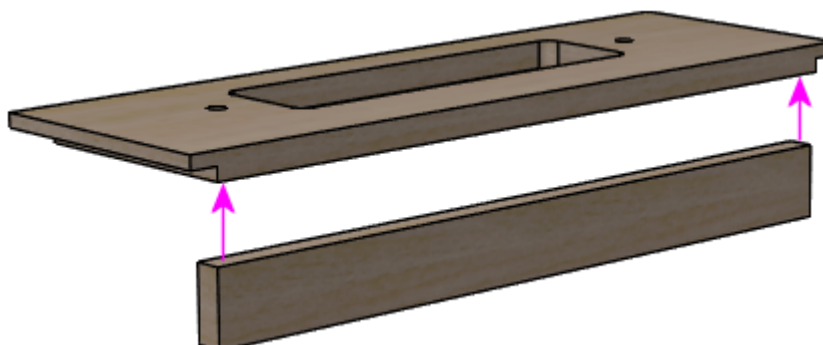
It's also great to have an assistant on hand. The job is easier with two people.

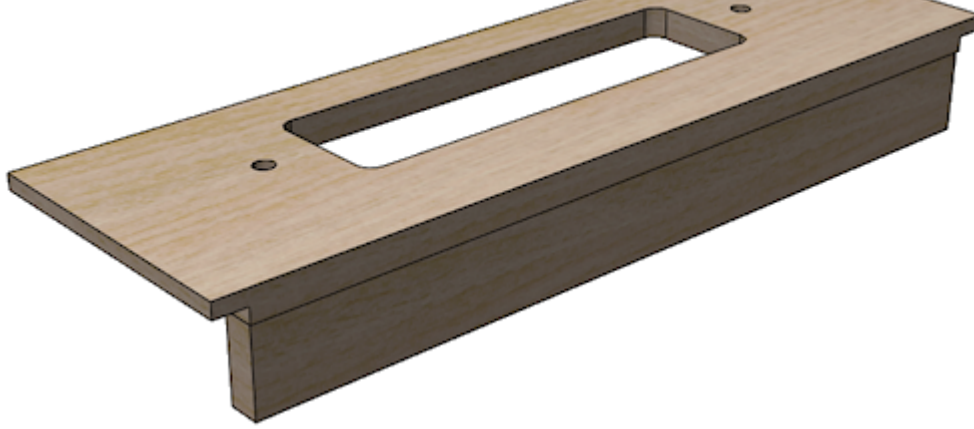
It should be fairly obvious how the pieces fit together, but here's a suggested assembly order.

Pre-assemble the shelf

Install a #6-32 x $\frac{3}{8}$ " on the bottom side of each bolt hole. (These are there to mate with safety bolts screwed in through the matching holes in the backbox, to secure the backbox in the upright position.)

Glue together the two pieces that make up the shelf as illustrated below. The front edge of the lip should be flush with the front edge of the top piece.





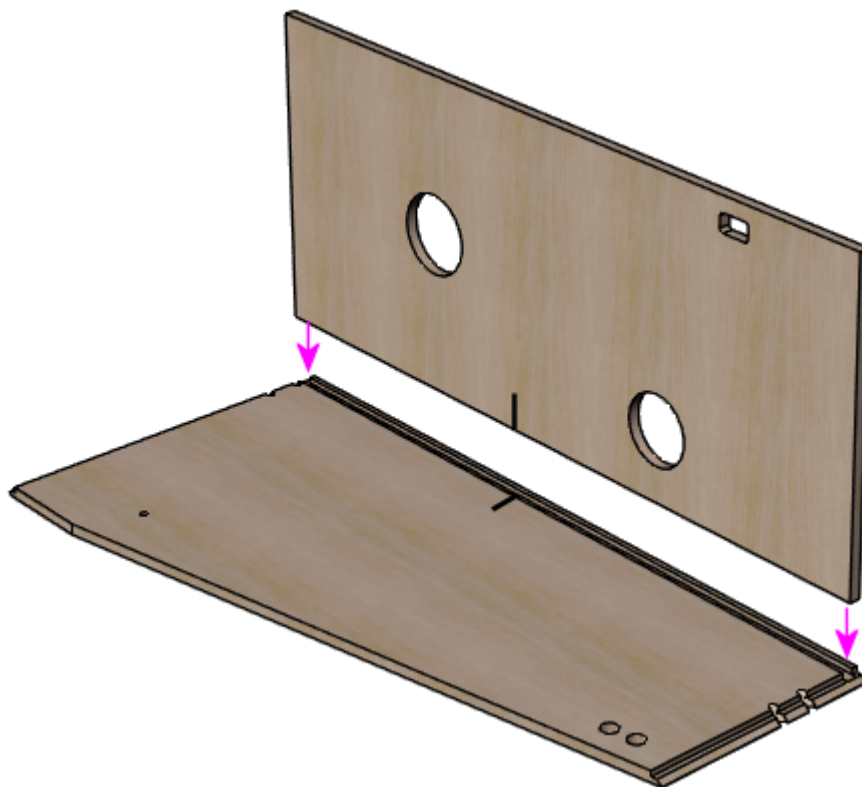
Set the assembled shelf aside for the glue to dry, so that it'll be set when we're ready to install it in the cab later on.

Main cabinet

Start by joining the floor to one of the side walls. Put glue along the inside of the dado (groove) at the bottom of the side wall as illustrated below. Don't use an excess of glue - you just want a single continuous bead down the center of the groove. Insert the floor into the groove. Make sure the front and rear edges are properly aligned and flush, and ensure that it's pressed down all the way into the dado.

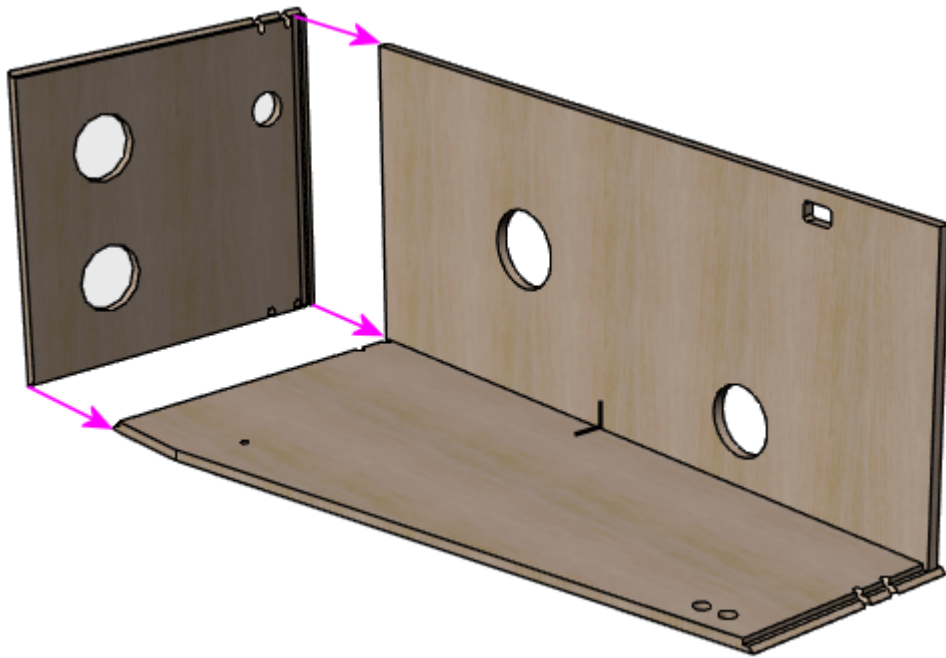
If you made alignment marks during the dry fit, this is the time to use them! Make sure that the alignment marks made earlier on these two pieces line up. That should ensure that the floor is properly aligned relative to the front and back pieces later on.

Beware that this arrangement is precarious! The floor piece will want to tip over; the dado isn't strong enough by itself to hold it upright. Keep the floor piece supported so that gravity doesn't stress the joint. It's good to have an assistant to hold things in this position until you get to the next piece.

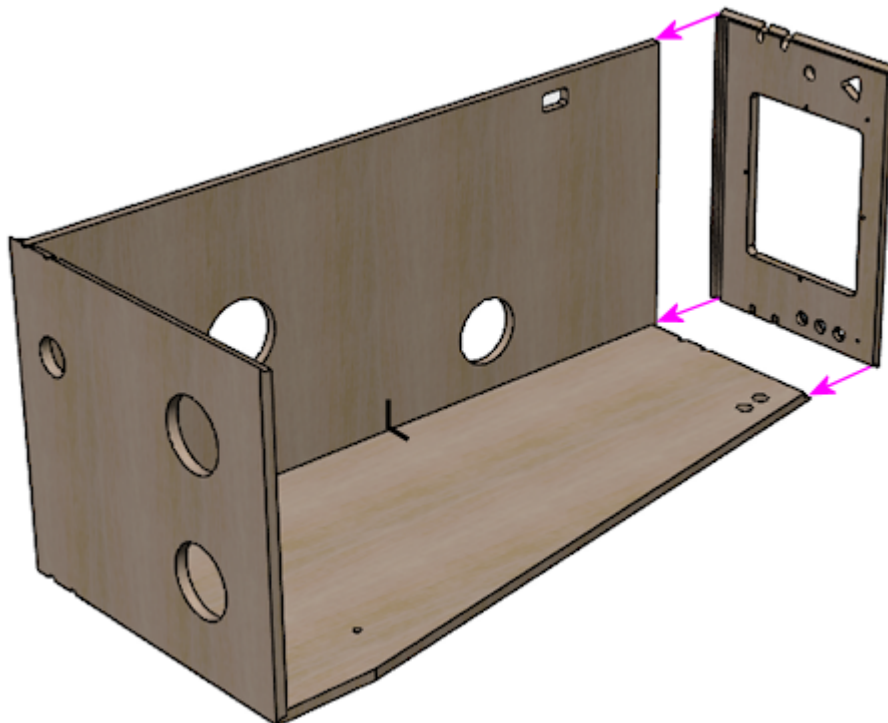


Next, add the back wall. Put glue along the dado and edge of the back wall that

we're about to join, as shown below. Again, use continuous bead of glue. Put the back wall piece in place. As before, make sure that the edges are aligned properly and that the floor is pressed all the way into the dado in the back wall.

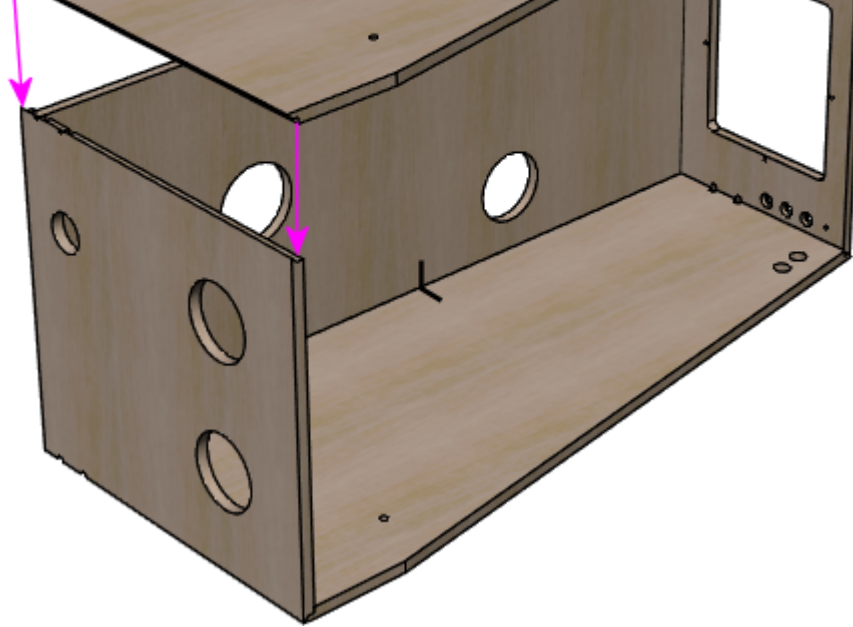


Now do the same thing with the front wall.



Add the remaining side panel.





Leg brackets

The next step is to install the leg brackets. The brackets will be permanently installed in the cabinet, so this is a one-time step that you won't have to repeat when you want to attach or remove the legs.

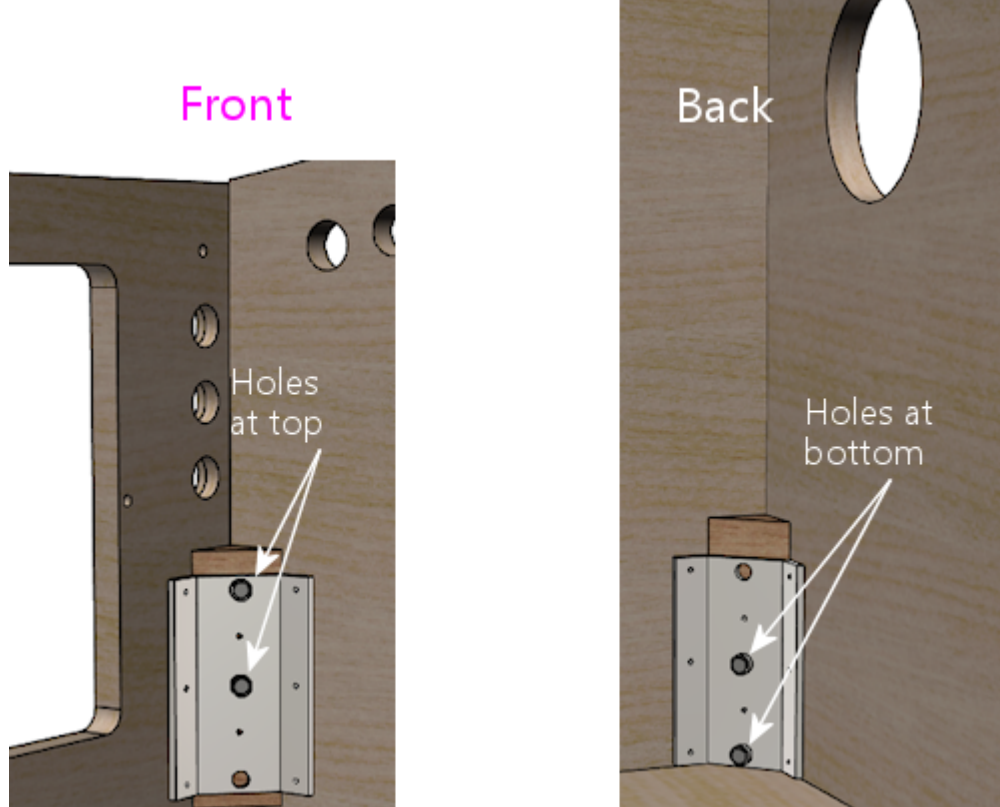
The procedure here assumes you're using the standard brackets used on newer machines, Williams/Bally part 01-11400-1. These brackets have integrated threading for the bolts, so no additional nuts or other fasteners are needed - you just screw the bolts into the brackets.

You'll need four of these brackets. The matching bolts are $\frac{3}{8}$ "-16, in $2\frac{1}{2}$ " or $2\frac{3}{4}$ " lengths. Note that you'll probably want to buy the bolts from a pinball vendor rather than use generic hardware store bolts, for cosmetic reasons: the ones made for pinball machines have nice shiny finishes and rounded heads that look nicer than generic galvanized hex-head bolts. You'll need eight bolts (two per leg). No washers or nuts are needed, as the brackets are threaded and serve as the fasteners.

The recommended brackets have their own threading for the bolts, which lets you attach and detach the legs purely from the exterior of the cabinet. In other words, there's no need to reach inside the cabinet with a wrench to turn a nut or other fastener, since no other fasteners are needed - the bolts screw directly into the threaded holes in the brackets. That's important because it's difficult to reach into the interior corners (especially with a wrench) once all of the equipment inside is installed. So the threaded brackets make things much easier in the long run, but they require some extra work for the initial installation, since you have to align them and fasten them inside the cabinet. That's what the procedure below is intended to accomplish.

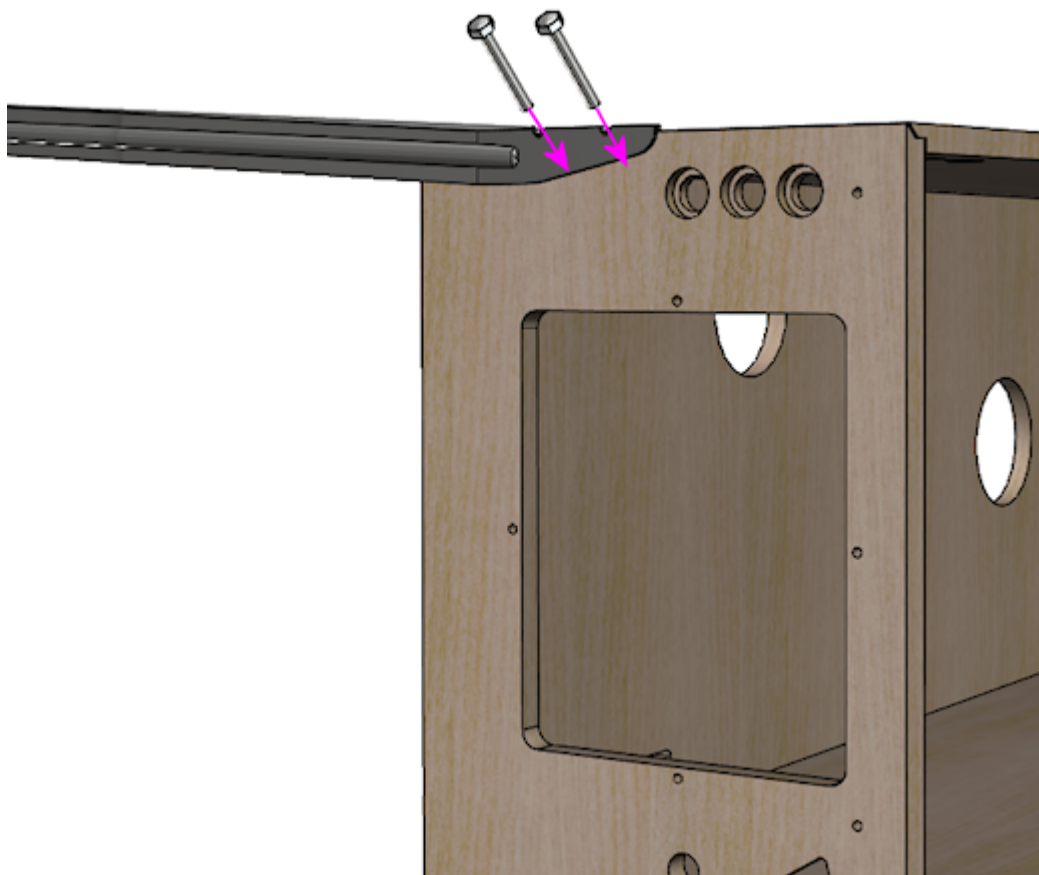
All four legs are interchangeable - there's no such thing as a "front leg" or a "back leg" or a "right leg" or a "left leg". You should simply have four identical parts for the legs. The same is true of the metal leg brackets.

Before we begin, it's worth noting how the positioning of the leg bolts relative to the floor of the cabinet affects how the brackets and corner braces are installed. The bolt holes are higher up on the wall in front, lower in back, to give the cabinet a slight forward tilt when it's set up. (The legs themselves are all the same length, so we get the tilt by mounting the legs at different heights.) Because of this asymmetry, we can flip the brackets upside down in front to keep them lower on the wall.



We'll start with a dry fit (no glue) to make sure everything fits, before we finalize the install. The bolt holes tend to be tight, which is good in that you don't want a lot of play or wobble when the legs are attached. But the bolt holes in the wood can be so tight initially that the leg bolts just won't fit. We need to make sure that the bolts will fit properly.

With the cabinet on its side, place the leg in position, and insert the bolts through the leg holes and into the cabinet. If the fit is too tight to get them through by hand, use a round file to ream out the holes enough to get them to fit.

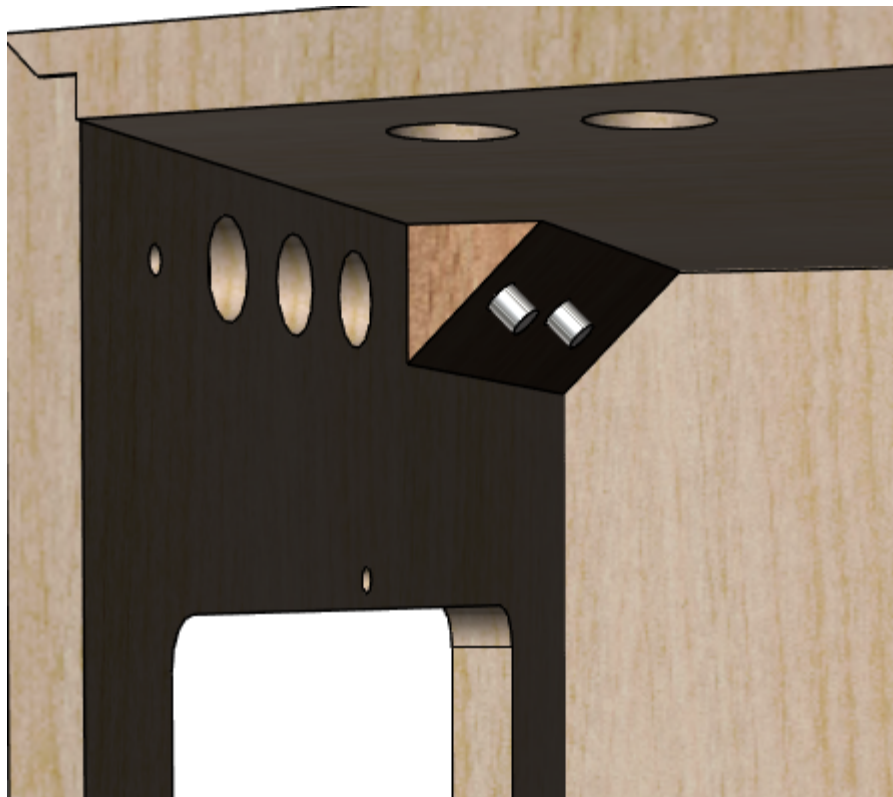
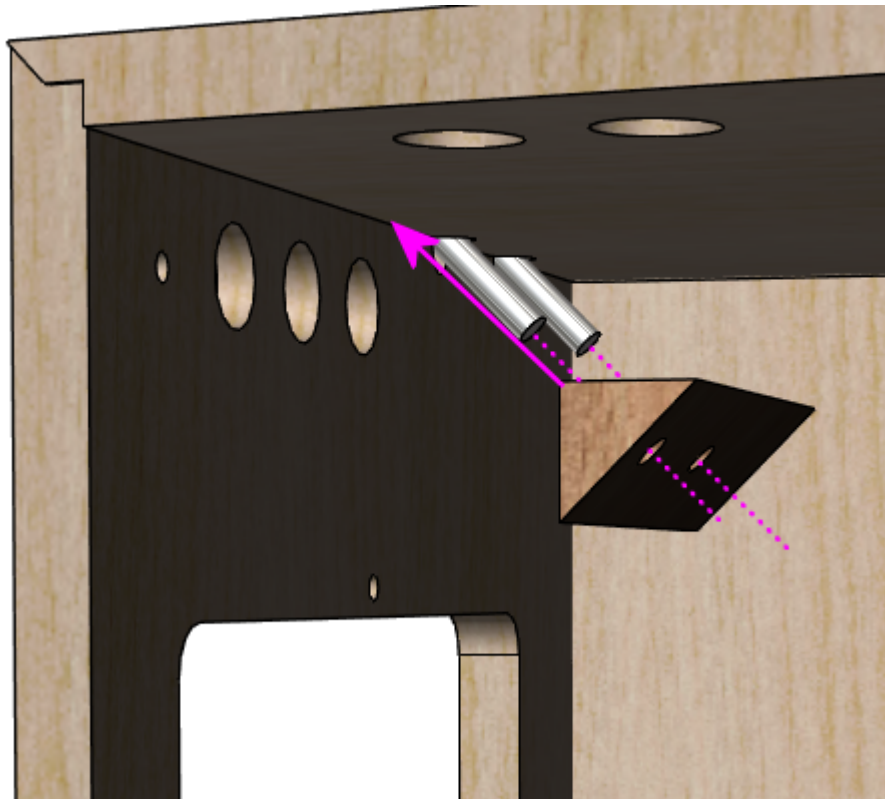




The point of using the legs for this step is just to make sure that the spacing of the bolt holes in the legs matches the spacing in the cabinet. We're not actually attaching the legs permanently yet; we're only attaching the brackets at this point. The legs can be easily attached and detached at any time once the brackets are

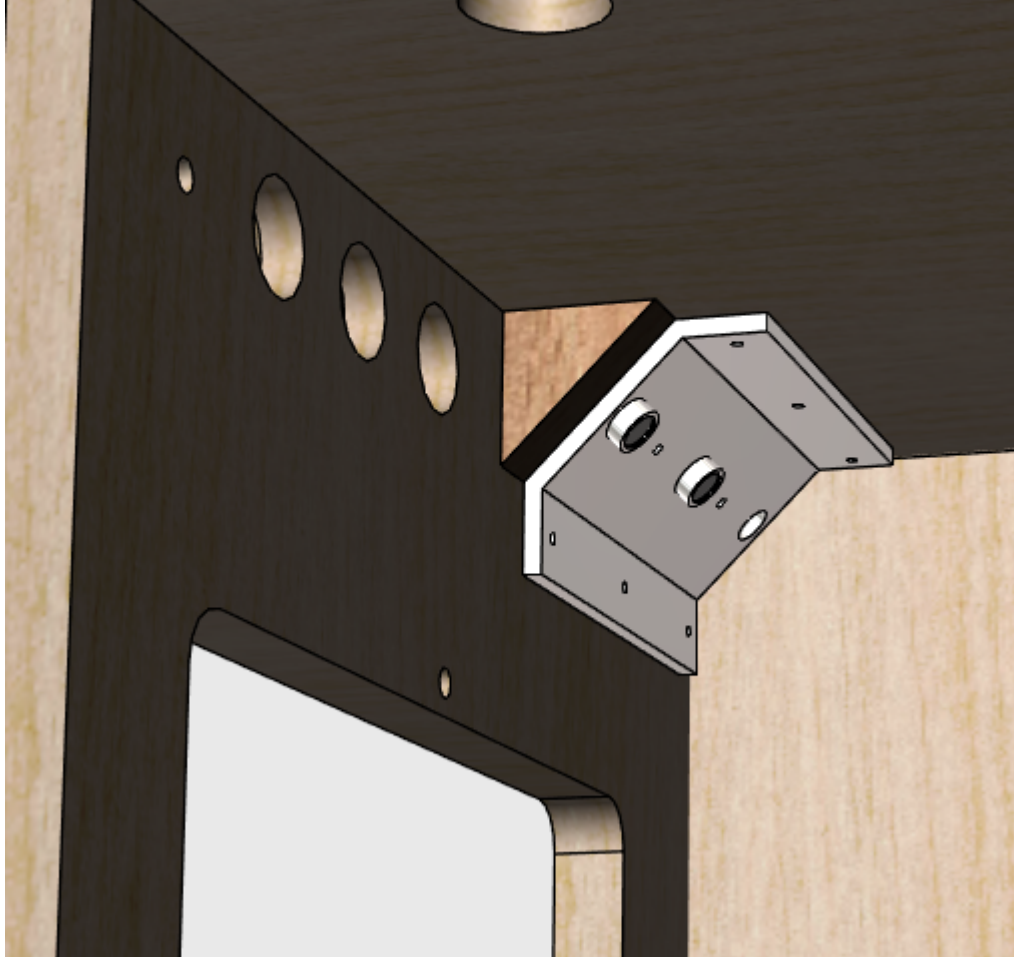
installed.

Once the bolts fit comfortably, slip the triangular wood space piece over the bolts.



Now attach the metal leg bolt bracket. Screw in the bolts to make sure everything still fits.





If anything is wrong with the fit, go back and use a round file to open up the holes in the cabinet walls and/or the corner braces as needed. (Obviously, don't attempt to modify the legs themselves or the metal bolt bracket! We consider those to be the source of truth here - they're the reference points we're trying to match with the wood parts.)

Once you're satisfied with the fit, take the bracket off and remove the corner brace. We're now ready to install this all permanently.

Keep the legs and bolts in place, since we still want them there as the reference point for final alignment.

Apply glue to the sides of the corner brace that face the cabinet walls. (Those are the narrower sides. Don't glue the wider side that faces the bracket.) Use a thin layer of glue covering the whole face. Avoid the area around the bolt holes to avoid too much glue oozing in there.

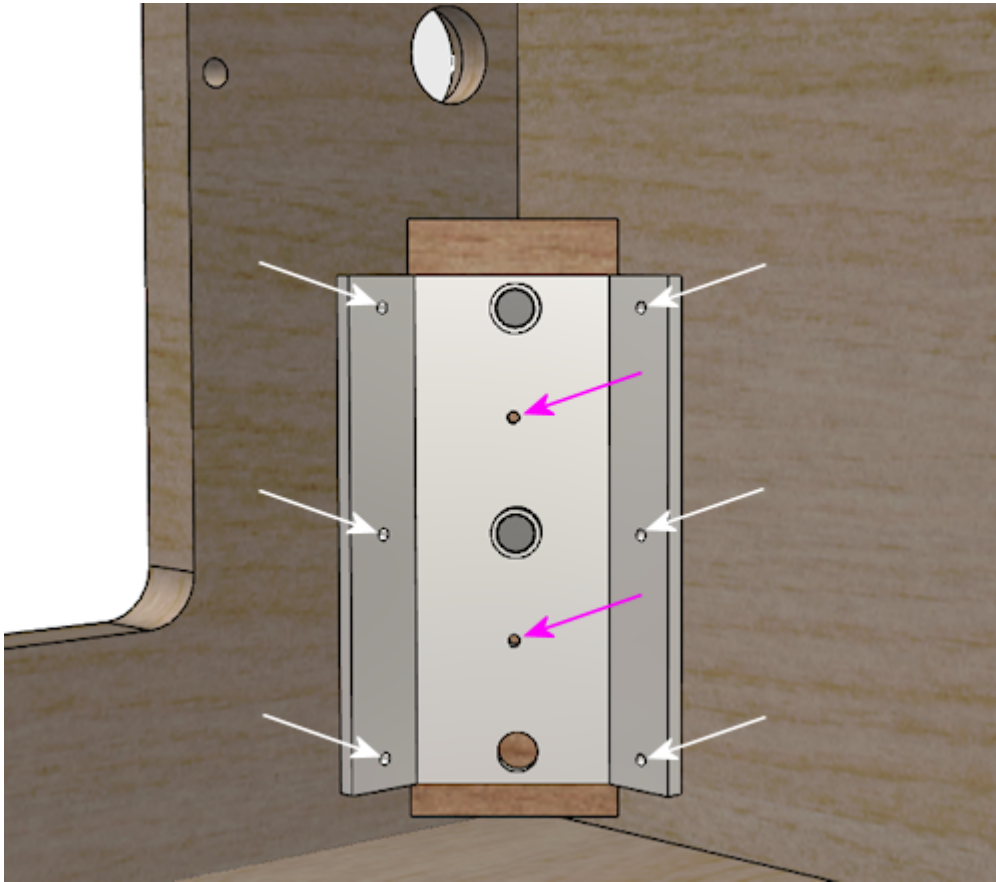
Put the corner brace back in place. Press it against the cabinet walls to attach the glue.

Reattach the bracket and screw the bolts into it. Screw them in all the way this time so that the leg is firmly attached. Don't over-tighten.

Use **#8 x 5/8" wood screws** to attach the metal bracket to the cabinet walls and to the corner brace. The standard plates have holes for three screws on each side and two more in the middle to attach to the brace. Don't leave out any screws; we want the bracket attachment to be very sturdy, so we want to distribute the load over as many screws as possible. Tighten the screws but be careful not to over-tighten and strip the wood.

Note: some people recommend using #10 x 3/4" screws instead of #8, since the

larger screws are a bit stronger. I don't think it's necessary - the original Williams cabs used #8 screws, and those seem to hold up over the years - but I also don't see it doing any harm to upgrade to the larger screws. The only thing to watch out for is that 3/4" screws could potentially poke through 3/4" plywood, or could come close enough to poking through to dimple the outer veneer. Check before screwing them in that they're not too long for your actual plywood stock. If it looks like it's going to be close, you can simply add a washer or two to each screw for some extra padding.



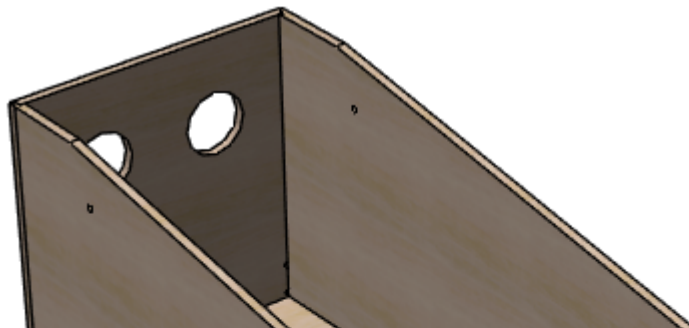
Use #8 x 5/8" wood screws to fasten the leg bracket to the cabinet and corner brace at the locations shown (arrows). You can substitute #10 x 3/4" wood screws for greater strength if desired, but check them against your plywood stock to make sure they won't poke through the other side.

Once the wood screws are all in place, unscrew the main leg bolts and remove the leg.

Repeat this process for each corner until all four leg brackets are attached.

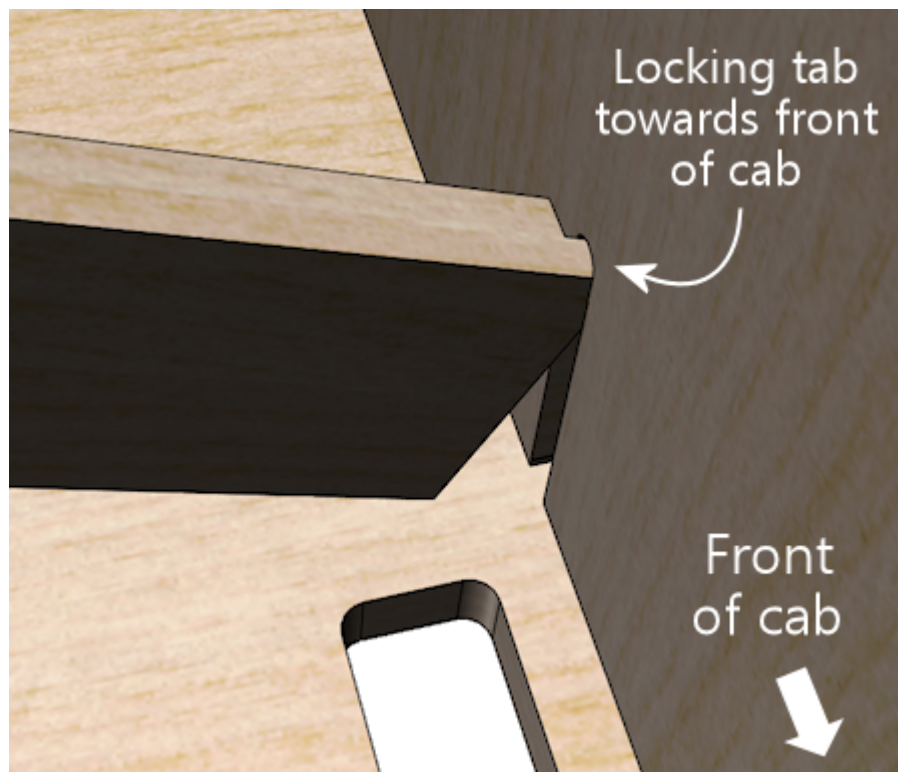
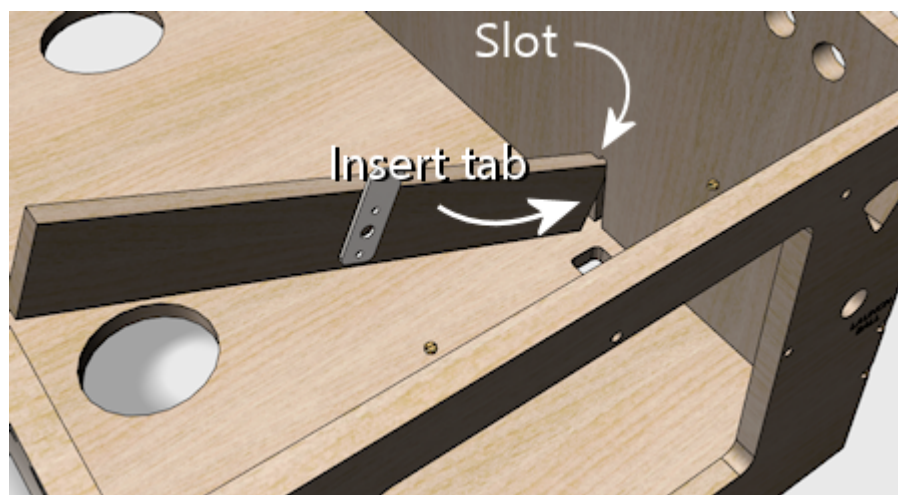
Cashbox fence

If you decided to include the fence that delineates the cashbox area, this is a good time to install it. Flip the cabinet upright for this step.

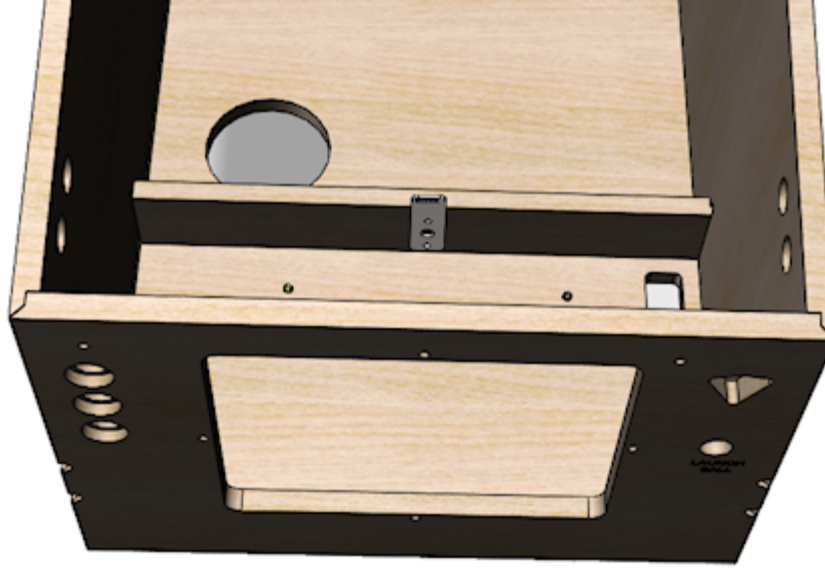




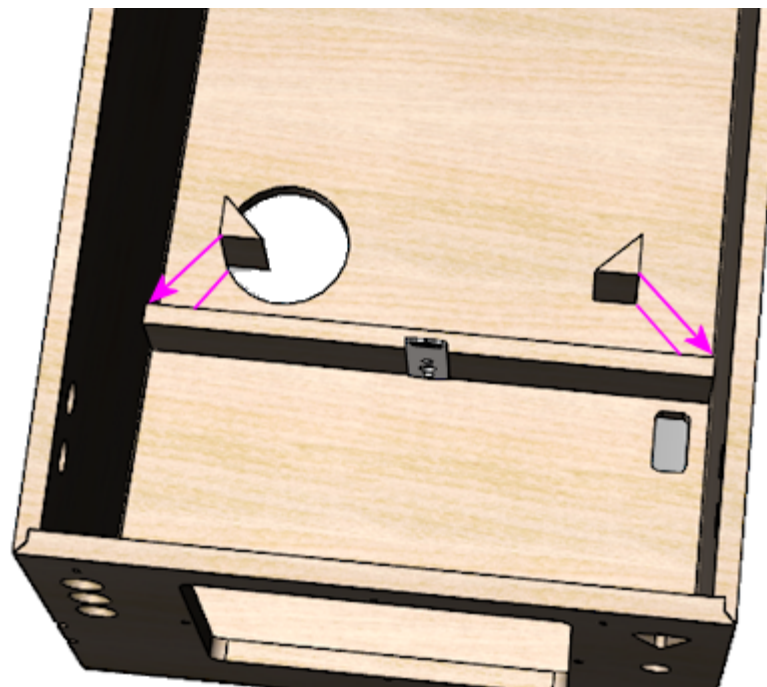
Installation option 1: using the locking tab. If you routed a slot in the side wall for the fence's locking tab, apply some glue to the tab and to the bottom of the fence, and fit the tab into the slot. Orient the piece so that the tab is on the side facing the **front** of the cabinet.



Place it against the floor, straight across the width of the cab.



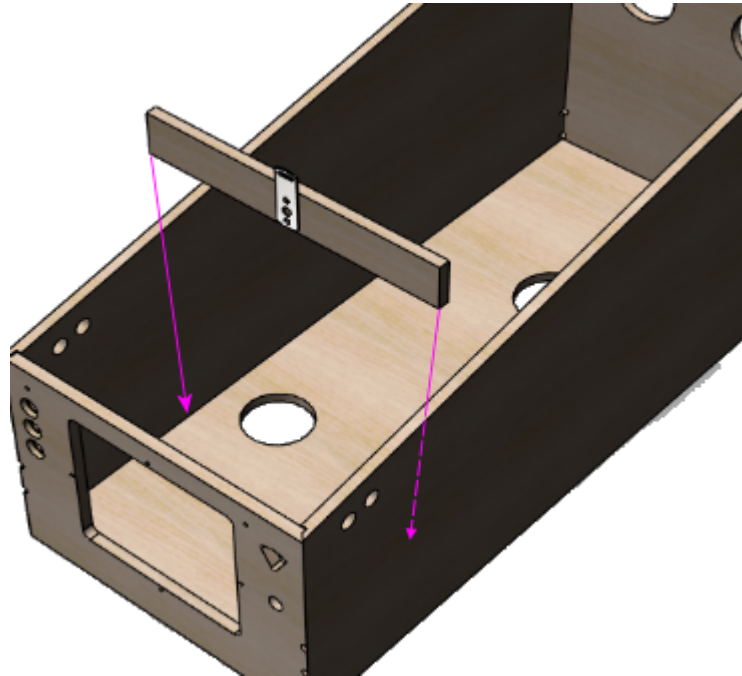
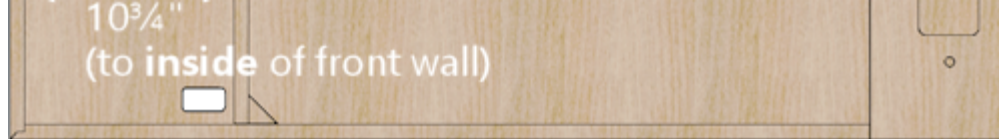
Glue the triangular braces behind the fence at the corners.



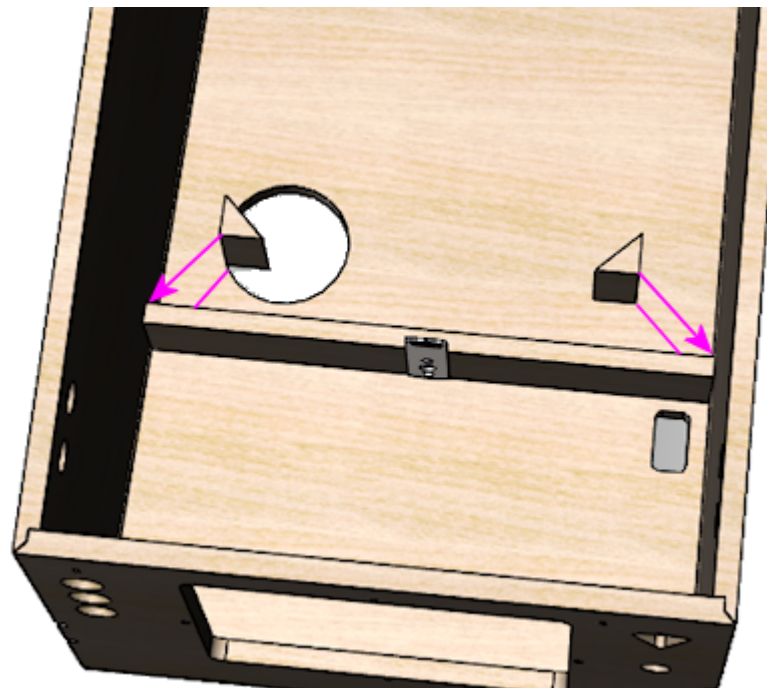
Installation option 2: no locking tab. If you chose to skip the locking tab and slot, you can position the fence now to fit your cashbox. Grab your cashbox and place it against the front wall. Position the fence to leave about 1/2" of play behind the cashbox.

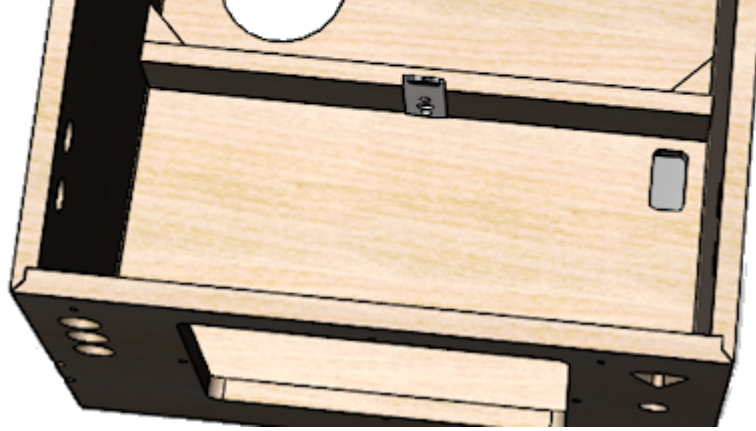
Without using any glue yet, set the fence in place at the desired position. The normal location, to fit a standard cashbox, is 10-3/4" back from the inside of the front wall, but you should change this to match your cashbox's depth if you're using a non-standard size.





Apply glue to the two square sides of the 3"-tall triangular pieces that you cut along with the fence. Making sure to keep the fence at the desired position, press the triangular pieces into place on the rear side of the fence at each side, to fasten the wall to the two sides of the cab.



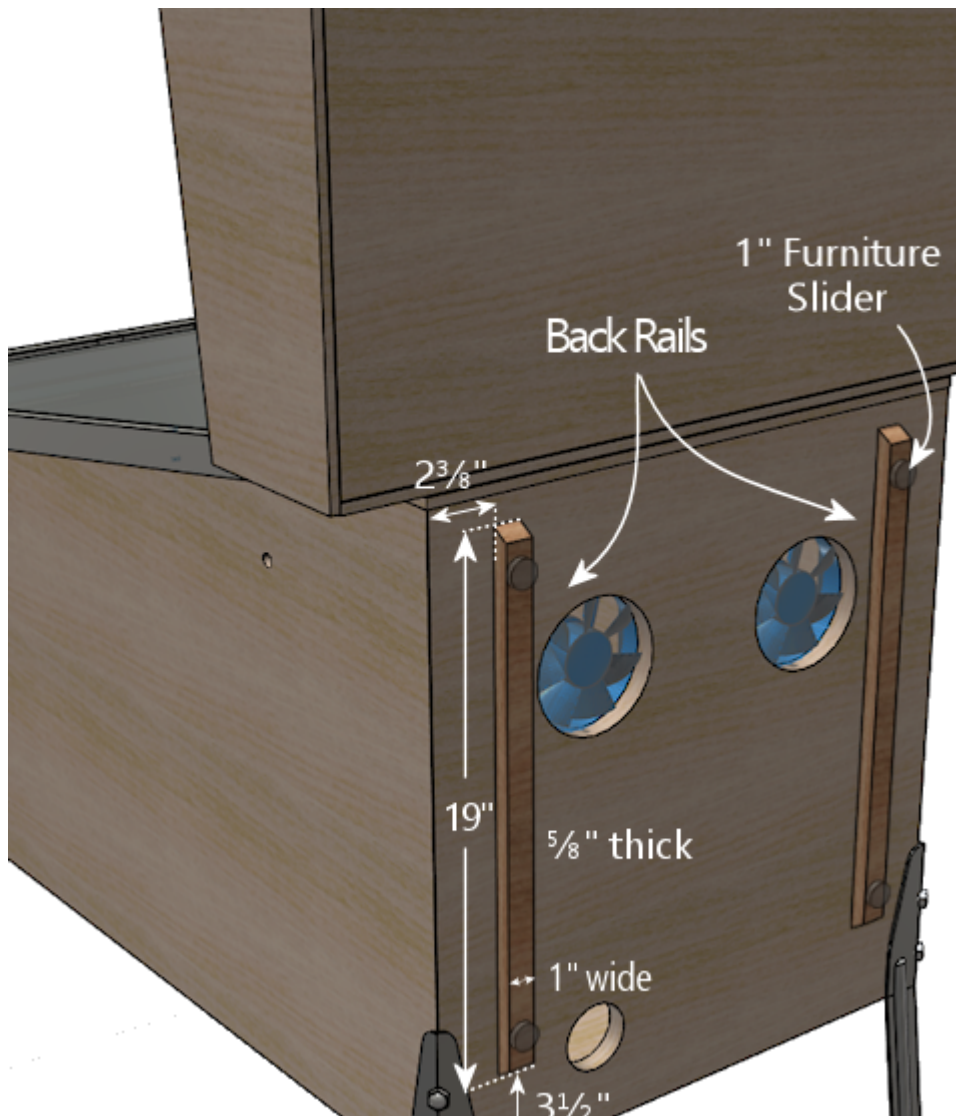


Back rails

If you want to include the back rails, attach them to the back of the cabinet, oriented vertically, near the edges. The exact positioning isn't particularly important, as long as the rails form a stable base for standing the machine on its back, so make any adjustments needed to keep clear of your fan vents and other openings in the back wall.

Attach these to the back with glue and finish nails (1¼" #18 brads should work). Nail down the centerline, with a nail every 4" or so.

If desired, affix hard plastic furniture slider pads near the ends. The exact type isn't important; the ones Williams used in the 1980s and 1990s were typically the nail-in type, hard plastic, ¾" diameter, white or tan.





Shelf

At this point, you can install the shelf that you assembled back at the start of the build process. We saved this for last, because the shelf gets in the way when you're trying to work around the back of the cabinet interior (such as installing the rear leg brackets). In fact, for just this reason, you might want to wait even longer on this step, and come back to it later, after installing parts that attach to the back wall, such as:

- Fans
- Power inlet
- Power strips
- Ethernet port
- USB ports

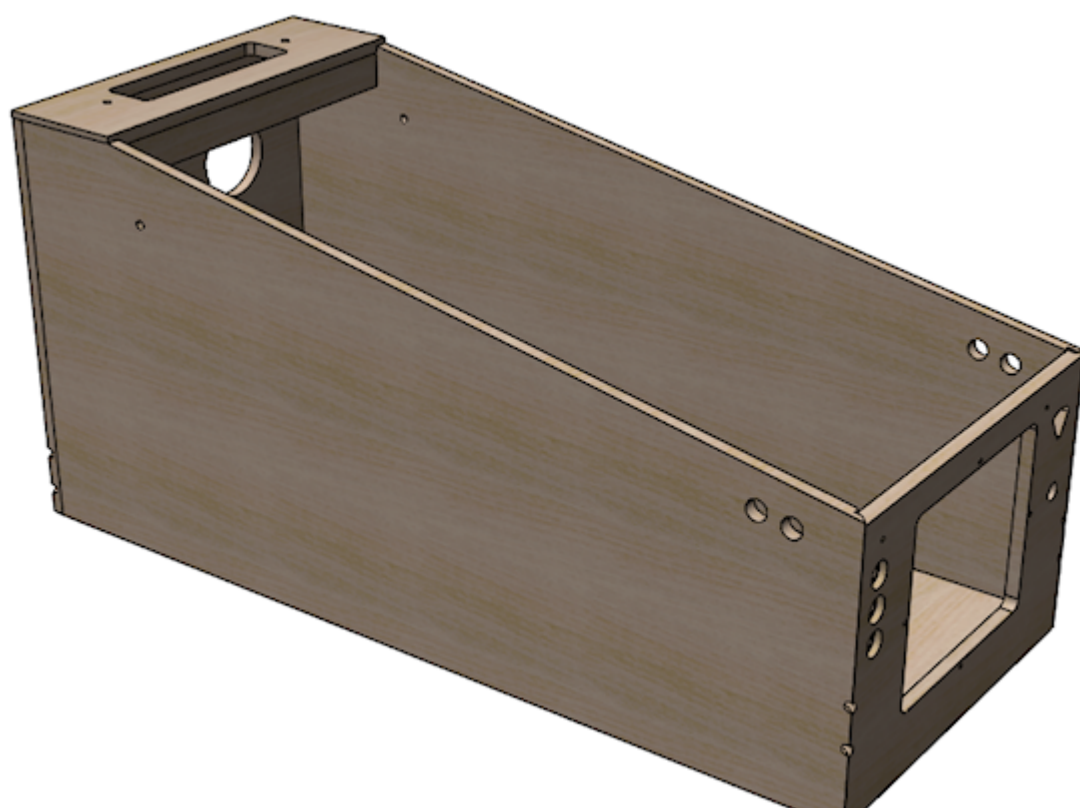
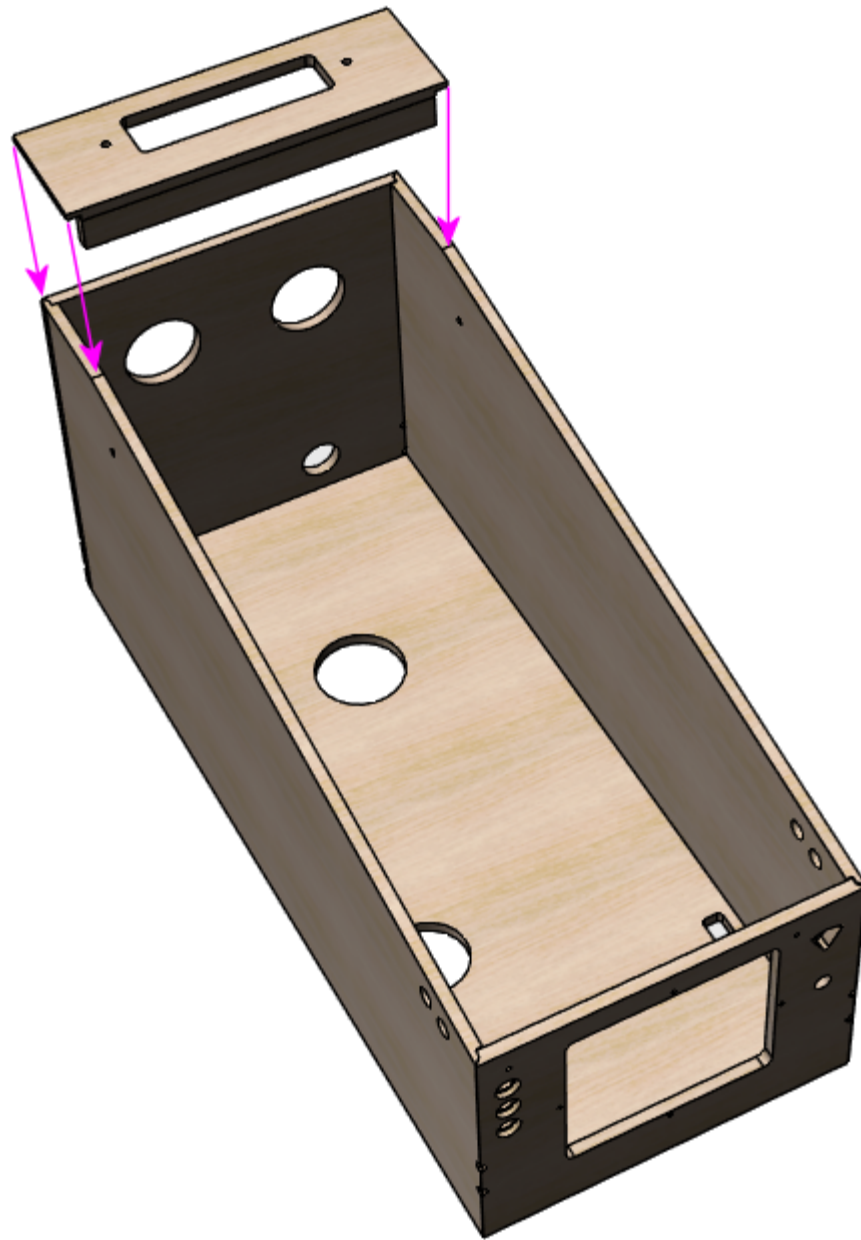
If you want to hold off installing the shelf for now, you can just set it aside and make a mental note to come back here when you're ready.

If you haven't already done so, install $\frac{3}{8}$ "-16 T-nuts in the holes on either side of the central rectangular opening, on the **bottom** side of the board. These mate with the wing bolts that are meant to be attached through matching holes in the floor of backbox. The bolts are an important safety measure to secure the backbox in the upright position while deployed.

Once you are ready to install the shelf, start by flipping the cabinet upright.



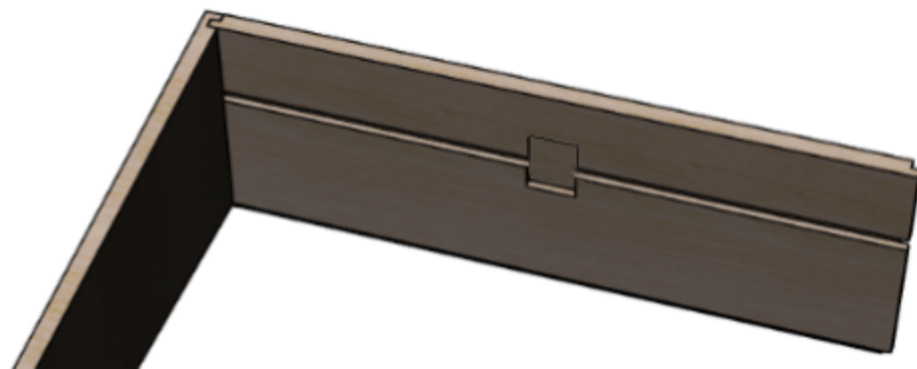
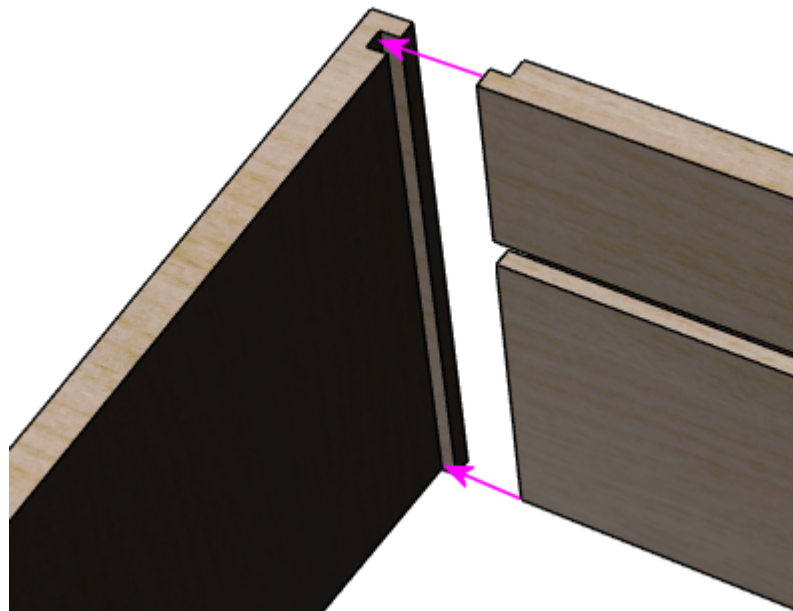
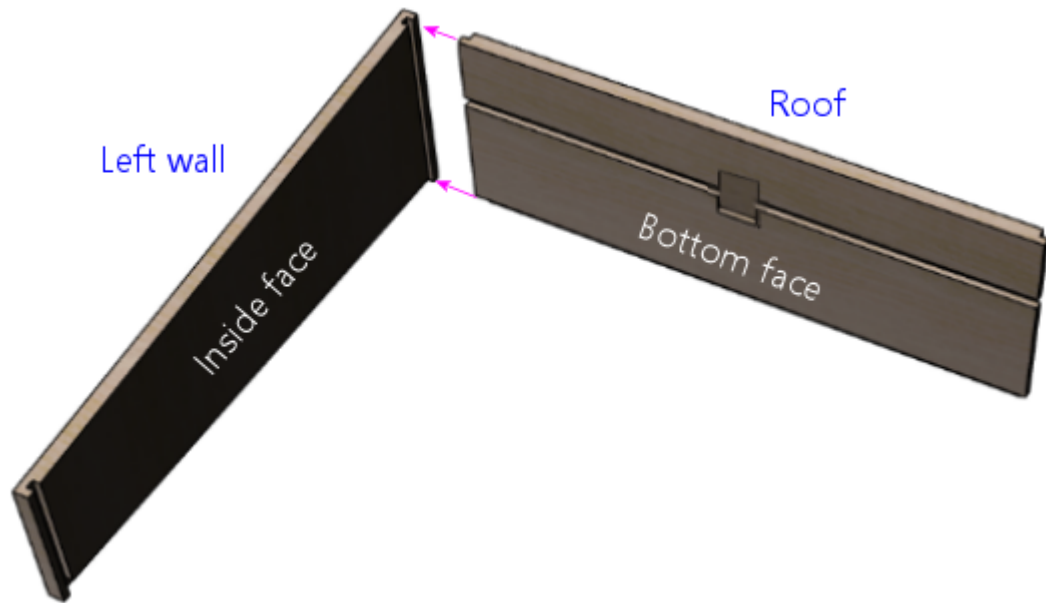
Run glue around the edges of the shelf where it joins the main cabinet (as shown below), and set it in place.



If the top of the shelf sticks out at all from the side or back walls, use a power sander to remove excess material until it's flush with the adjoining wall.

Backbox

Assembling the backbox is much like assembling the main cabinet. Start with the top and one of the side walls. Apply a bead of glue to the groove on the side piece, then fit the top piece into the groove.

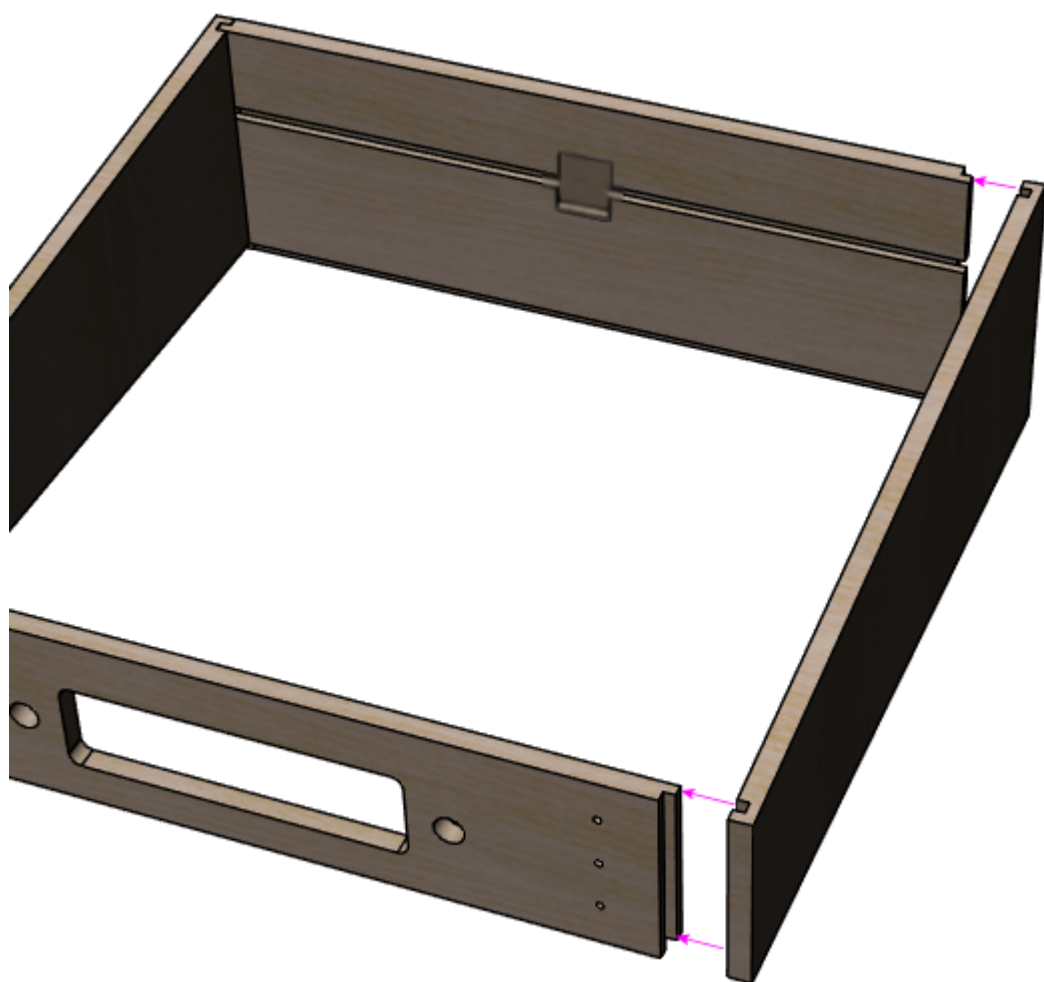




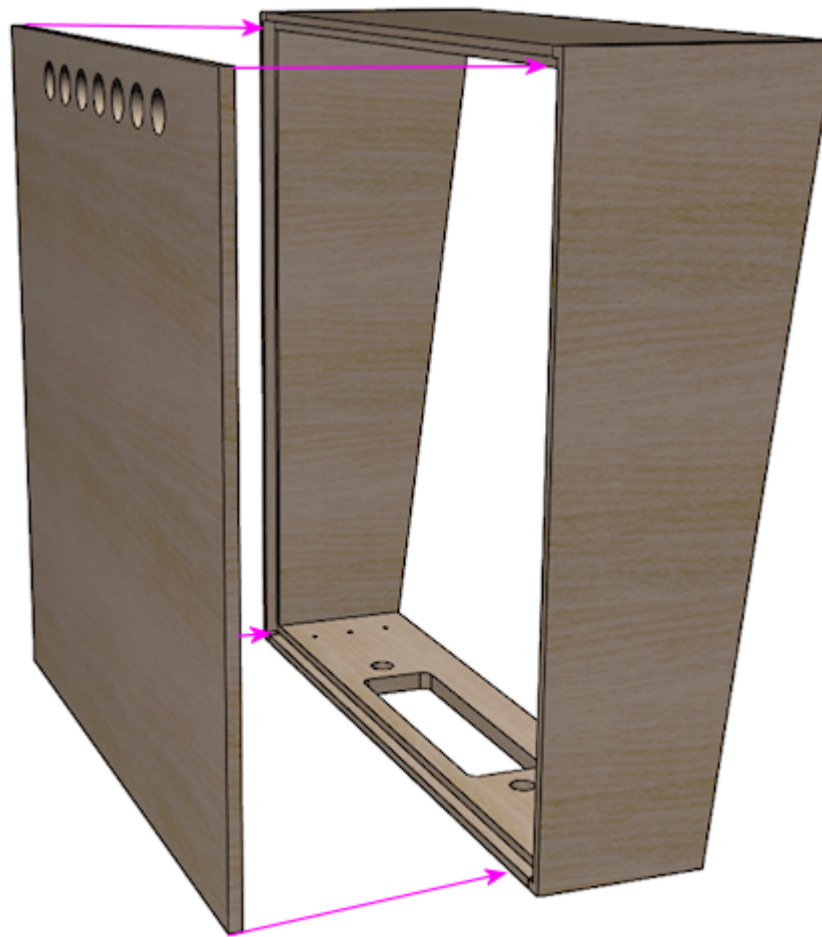
Attach the floor.

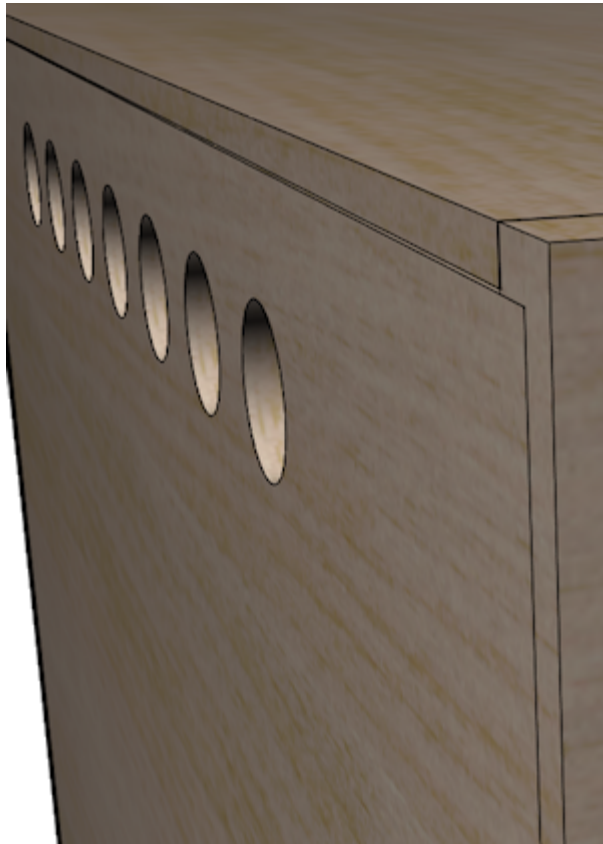


Add the remaining side wall.



The back wall should now fit into the grooves along the back edges of all four walls. Apply glue around the grooves, and put the back wall into place. It should fit so that it's flush with the edges of the walls.





The back should be flush with the back edges of the adjoining walls when installed.

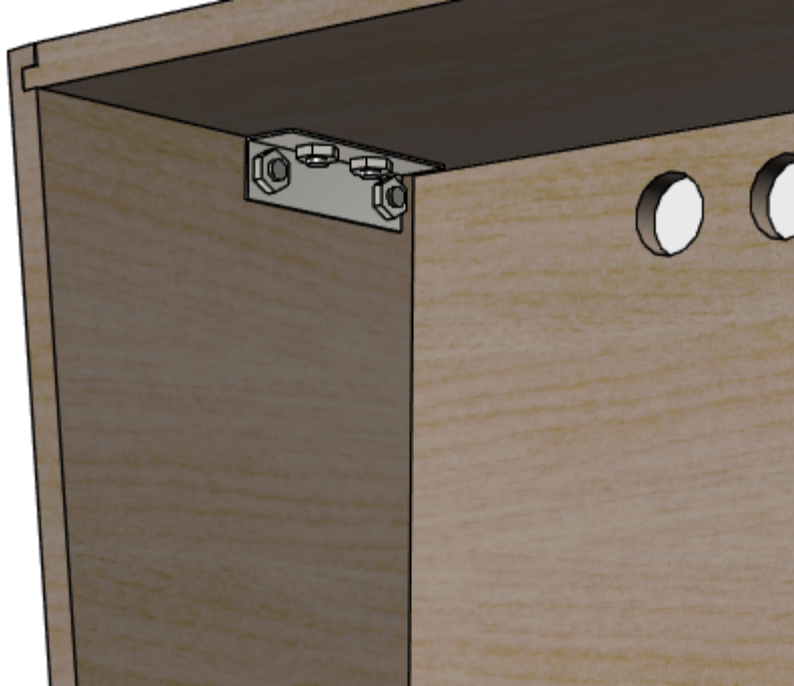
In addition to the glue, you can add some finish nails to strengthen the back wall. Use small finish nails, such as 1" #18 brads. Drive them in from the back of the back wall, around the perimeter, set in about 3/16" from the edges. Space them every few inches; four or five nails on each side should be sufficient.

Corner bracing

The original WPC backboxes had steel braces at the corners to strengthen the joints. The glued corner joints are actually pretty sturdy all by themselves, if you construct them using the rabbeted design described above, but apparently Williams deemed it necessary to add some heavy reinforcement. I'm sure that came from experiences with commercial operators who banged up their machines with rough handling and then complained when they broke.

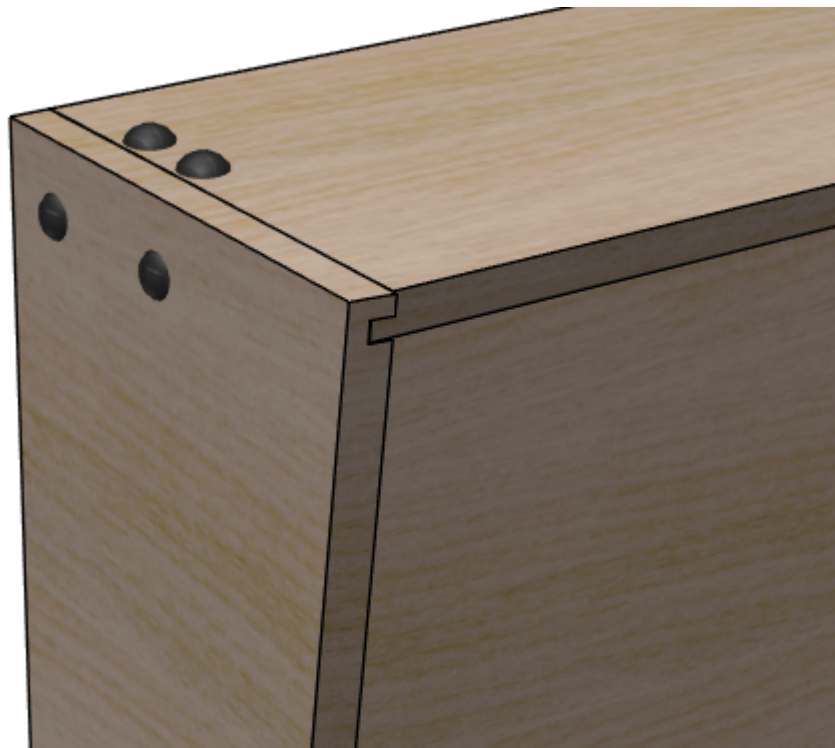
In my opinion, you shouldn't need any corner braces for a machine in home use. The glued corners should be plenty strong. But if you'd like to reproduce the original construction faithfully, or you just don't trust the glue joints, here are the details for the Williams design. The part number for the braces is #01-9167, and they're fastened to the backbox walls with 1/4"-20 x 1-1/4" carriage bolts (black finish, 4320-01123-20B) and 1/4"-20 flange nuts (4420-01141-00). You'll need four of the braces and sixteen each of the carriage bolts and flange nuts. Place one brace at each corner, more or less all the way back against the back wall, and use the holes in the brace as a drilling template to drill holes for the carriage bolts. Insert the carriage bolts with the heads on the outside, and fasten with the flange nuts on the inside.





WPC backbox brace, Williams part #01-9167, installed at the upper corner. The real WPC-era machines used one bracket like this at each corner. If you want to go this route, use the brace as a drilling template to drill $\frac{1}{4}$ " holes for the bolts, and fasten the brackets with $\frac{1}{4}$ "-20 x $1\frac{1}{4}$ " carriage bolts (on the outside) mated with $\frac{1}{4}$ "-20 flange nuts (on the inside).

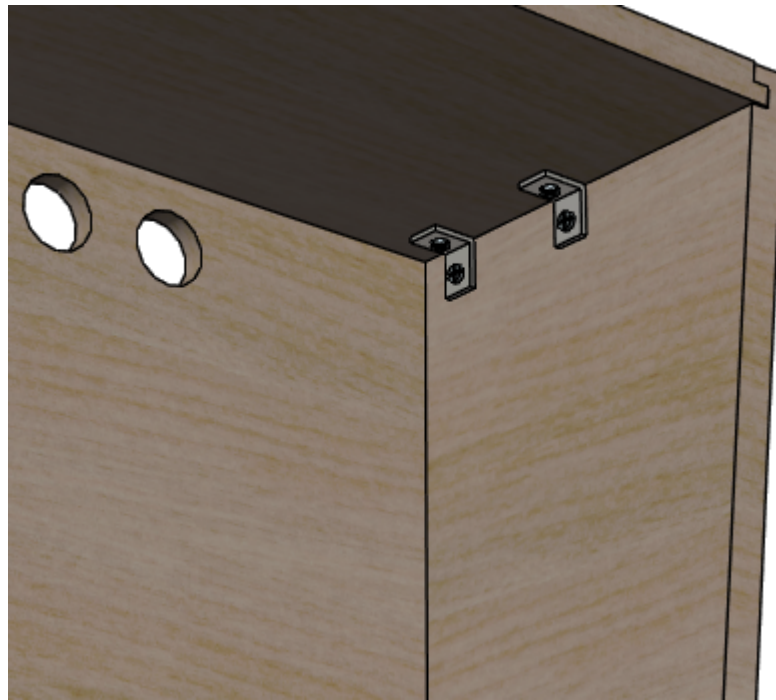
The Williams corner bracing is about as strong as you can get. You'd have to rip the wood apart before those bolts would come out. The downside is the bolts are visible on the outside of the backbox. (Not *too* visible, though; the WPC machines use black bolts that tend to disappear into the artwork unless you're looking closely.)



How the carriage bolts look on the outside. They have smooth rounded heads (with no screwdriver slots), and come in silver and black finishes.

If you don't care about using the exact original parts, but you still want some kind of

corner reinforcement, you might consider using generic steel 1" corner braces instead. You can buy these at any hardware store. Use $\frac{3}{4}$ "-long wood screws to attach them, in a size that fits the holes in the corner braces you buy (typically #6 or #8 wood screws). Use two or three braces per corner. Keep them within 5" of the back wall, so that they won't be visible when the translite is in place. This setup isn't as ridiculously strong as the Williams brackets and carriage bolts, but it provides some reinforcement, and it doesn't require any externally visible fasteners.



Alternative reinforcement using generic hardware-store corner braces, fastened with wood screws. Be sure to keep the braces behind the translite plane (5" from the back wall), so that they're not visible.

Translite/DMD guides

The WPC backbox has some little wood blocks along the walls that act as guides for the translite and DMD/speaker panel. These might or might not be interesting to you for your virtual cab, because a virtual backbox is a little different from a real one. Specifically, our backbox uses a TV in place of the normal translite, and in some cases a single TV replaces both the translite and DMD panel.

But it's not a simple matter of TV *or* translite. You might actually still want something similar to a translite, to mask out the bezel around the perimeter of the TV. There are two common ways to handle this:

- Create a custom wood cover for the TV area, with a cutout for the TV.
- Use a glass or plexiglass translate in front of the TV. Optionally, you can use paint or decals around the perimeter of the plexi to mask out the dead space beyond the edges of the TV display.

Both serve the same function, of hiding the TV's bezel so that you only see the screen, but I very much prefer the second option. The first option calls way too much attention to the virtual-ness of the cab. The second makes it look like a real pinball machine.

(There's a third, less common option. Some people route grooves into the side of the backbox exactly deep enough to contain the TV's bezels. This requires an extremely thin bezel, and requires that you use a custom backbox size chosen to perfectly

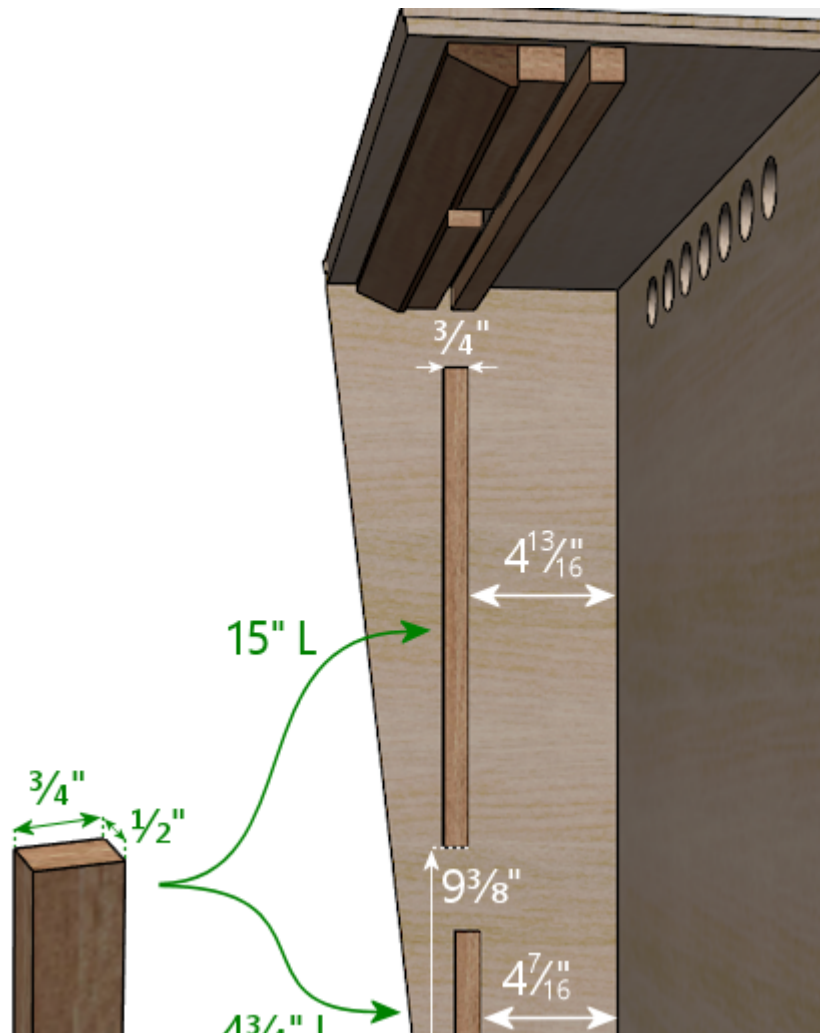
match the TV, so it's not compatible with the standard plans.)

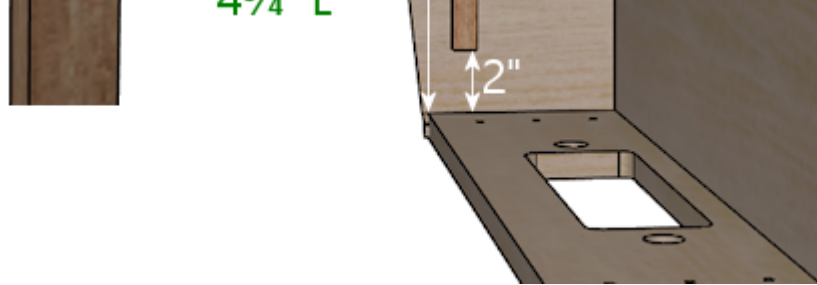
If you're planning to use a custom wood cover instead of a translite, you can skip this section, as your custom cover won't need the guides that hold the conventional parts in place.

Before proceeding with installing these, there are some cases where some of the guides should *not* be installed:

- If you're not using a standard speaker/DMD panel, don't install the lower guides (the ones at the bottom of the side walls) until you've worked out whether or not you need them. These are designed for the pre-WPC-95 style of speaker panel only, and might not work if you're using a home-brew design of your own.
- If you're using a WPC-95 speaker panel - the type that's made out of a single piece of molded black plastic - don't install the lower guides. The lower guides are only for the older pre-WPC-95 speaker panel. If you're using a standard panel type but you're not sure whether it's WPC-95 or pre-, consult Chapter 31, Speaker/DMD Panel for help.
- If you haven't finalized your backbox TV install plan yet, don't install the upper side wall guides. Those get in the way of some TV installation methods. See Chapter 30, Backbox TV Mounting for more.

Assuming that you're using the standard translite and the early 1990s style of speaker/DMD panel, here's a cutaway view showing the placement of the guides on the sides of the cabinet. Note that the right side wall isn't shown in this view, but (as you would probably expect) has the same two guide pieces shown on the left wall, at the same positions in mirror image.



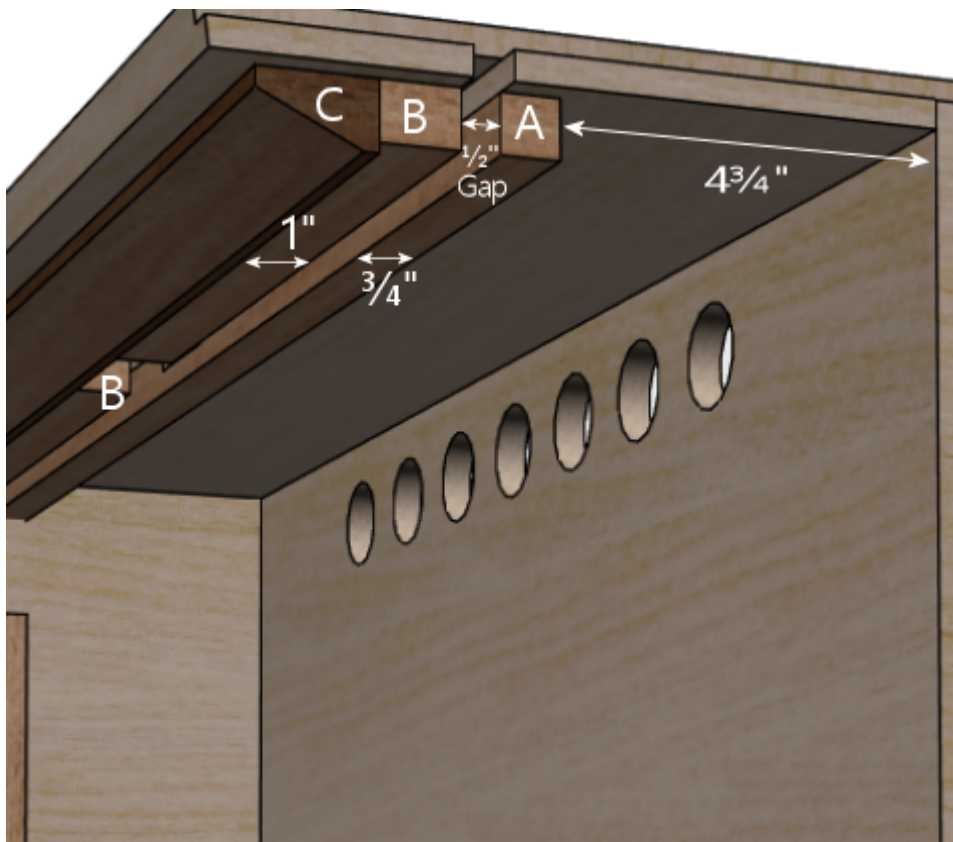


*Guides for the translite and speaker/DMD panel on side walls. The distances shown are to the inside surfaces of the back wall and floor in the assembled backbox. Note that some backbox TV installation designs work better **without** the 15" upper pieces, so you might want to defer installing these until you've finalized your backbox TV plan. Also note that the lower pieces are only used for the "original" style of speaker/DMD panel, **not** the WPC-95 molded plastic type.*

The top piece is 15" x $\frac{3}{4}$ " x $\frac{1}{2}$ ", and the bottom is 4 $\frac{3}{4}$ " x $\frac{3}{4}$ " x $\frac{1}{2}$ ". Orient them so that the $\frac{3}{4}$ "-wide face is against the side wall. Both pieces run parallel to the rear wall.

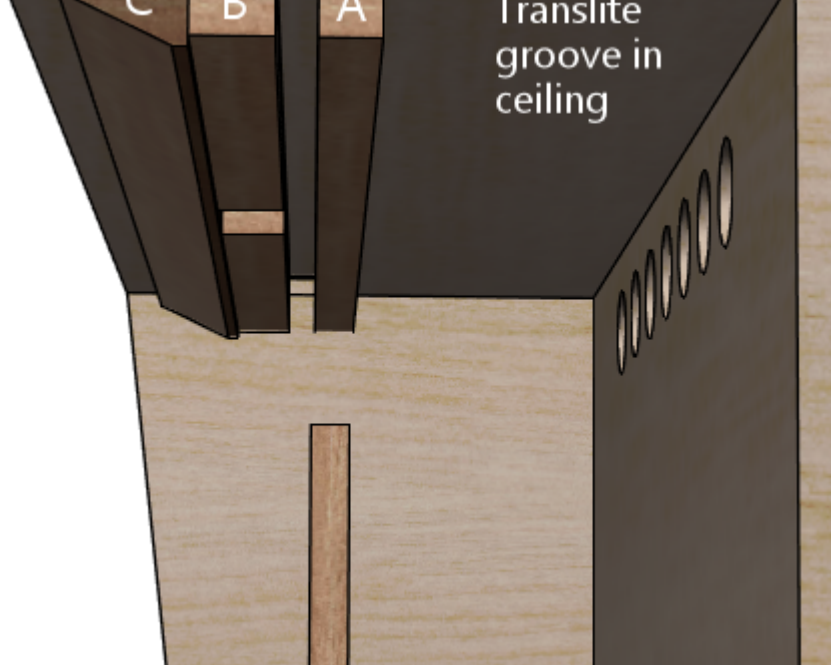
There's nothing sophisticated about the installation of these on the real machines - they're just glued and nailed. You should do the same thing. Apply a little glue on the back of each piece and nail it into place with finish nails (I'd suggest 1" #18 brads). Use one nail about every 4" down the length of each strip, centered in the strip.

Here are the guides on the inside of the backbox "roof":

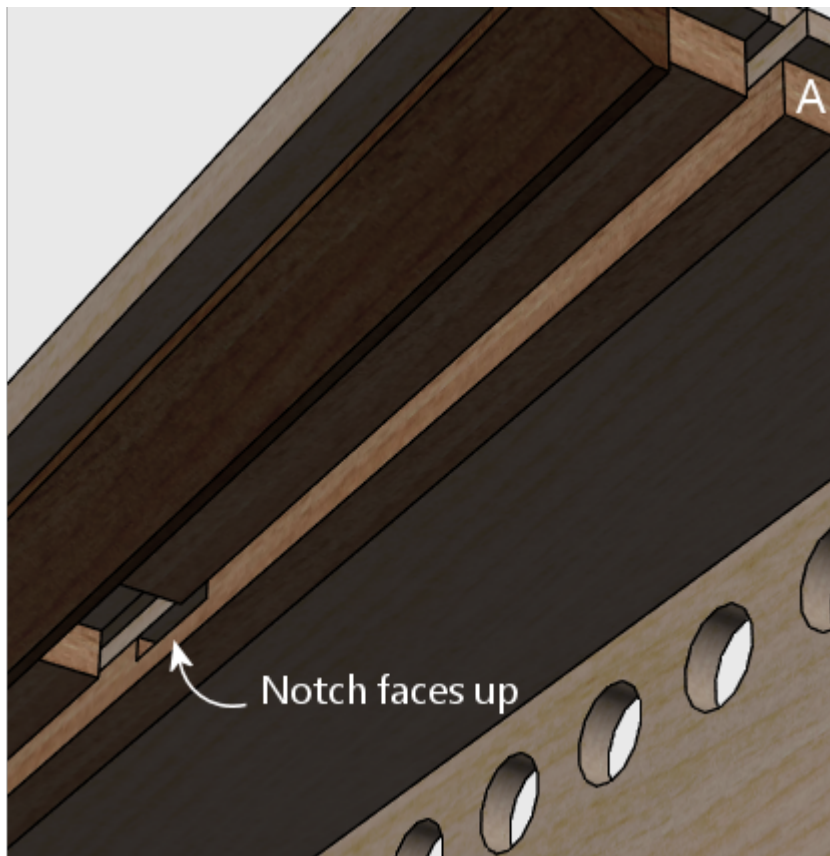


Cutaway side view of the top guides. The pieces labeled "A", "B", and "C" are detailed below.

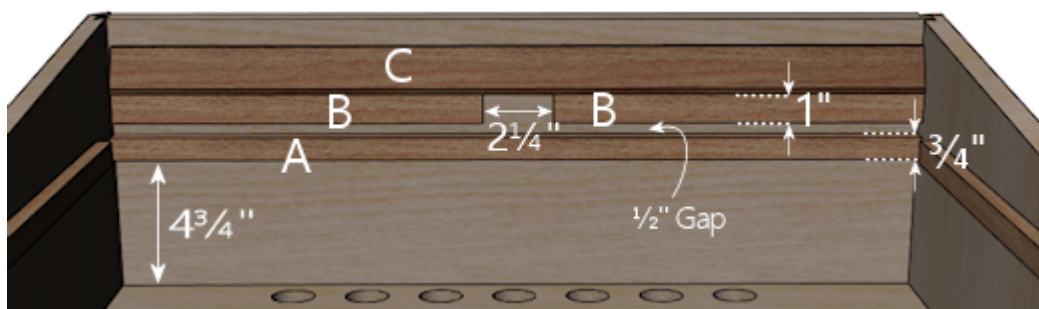




Note how the "A" and "B" pieces align with the translite groove in the ceiling.



Note that piece "A" should be installed with the notch facing the ceiling of the cab.





Top guides, viewed from below.

All three pieces run parallel to the rear wall.

- "A" is $27\frac{1}{8}$ " x $\frac{3}{4}$ " x $\frac{3}{4}$ ". It should fill roughly the full width of the backbox interior; simply center it left-to-right relative to any leftover space. Install it with the routed notch for the translite lock (if you included that) facing up, towards the ceiling of the cab.
- "B" (quantity 2) are each $12\frac{3}{8}$ " x 1" x $\frac{3}{4}$ ". Leave a $\frac{1}{2}$ " gap front-to-back between these pieces and the "A" piece, and leave a $2\frac{1}{4}$ " gap left-to-right between the two "A" pieces.

Important! If you installed T-nuts for the translite lock plate, see below.

- "C" is the wedge-shaped trim piece we described earlier. Orient it as illustrated in the side cutaway view above. Install it abutting the "B" pieces, without any gap.

Aligning the T-nuts in the "B" pieces: If you installed the T-nuts for the translite lock plate as described earlier, you should make sure they're correctly aligned for your lock plate when you install the "B" pieces. Use this procedure:

- Lay out the pieces at the install location as described above, but don't glue anything yet.
- Orient the pieces so that the T-nuts are on the side that will be glued to the ceiling of the backbox.
- Grab your lock plate and its Torx screws. You don't need to assemble the rest of the parts yet, but it's also okay if you've already done so.
- Put the lock into position. Make any adjustments to the positions of the "B" pieces to match up the screw holes in the lock plate with the pre-drilled holes in the "B" pieces.
- Fasten the lock plate by screwing in and tightening the screws.
- With the lock plate installed, glue and nail the trim pieces into position.

You can now remove the lock plate if desired (for example, if you still need to paint the backbox), knowing that the "B" pieces will be perfectly aligned for the lock plate when you're ready to re-install it.

22. Cabinet Art

If you're building your machine's cabinet from scratch, you'll want to decide on what the exterior will look like. This might be a simple flat black paint job, or you might prefer full-color graphics like on a modern real pinball machine.

Real pinball machines have always featured eye-catching cabinet artwork. The motivation was always commercial, of course - the art was there to grab your attention and entice you to drop in a few quarters. But that didn't mean it wasn't also art. Pinball has a recognizable graphics style - actually, several different styles over the decades, but each recognizably "pinball art". It's natural for virtual pin cab builders to want to tap into that by using artwork that would look at home on a real pinball machine.

Reproducing the authentic pinball art style can mean different things, depending on which era you're talking about. Machines built in the 1950s through 1970s tended to use abstract graphics, painted in three or four bold colors with stencils. The stencil artwork continued into the 1980s, but the graphics became more intricate and representational. In the 1990s, the manufacturers started using a multi-color silk-screening process, which allowed for higher-resolution graphics with more detailed designs.



Top left: Gottlieb's Abra Cadabra (1975), with abstract stencil graphics typical of machines built in the 1950s through 70s. Top right: Williams's Space Station

(1987), with the more intricate stencil graphics of the 1980s. Bottom: Bally's Theatre of Magic (1995), which used the high-resolution silk-screen graphics typical of the 1990s.

In the 2000s, the remaining manufacturers switched from screen printing to plastic decals. Decals are cheaper to produce, but they also offer more options to the designers, since they can be printed in high resolution and full color. (Silk-screening's palette is limited by the number of color layers used, and has to use half-tone patterns for in-between colors.) The switch to decals opened up even more options for art designers, including full photo-realism.

When to install artwork

I think it's best to paint or install decals just after completing the assembly of the wood cabinet, before installing any of the interior equipment, and certainly before installing the trim.

I'd wait until after assembly to do any decorating, because that lets you do a final pass with a power sander to even out surfaces, smooth corners, and remove any excess glue. It also eliminates the risk of scratching or marring the artwork during the assembly process.

It's better to paint and install decals before installing any interior equipment, since that will add weight and make the cabinet harder to move around. You'll want to be able to flip the cabinet onto different sides while working on paint or decals, so you don't want it weighed down with internal parts.

Virtual pin cab design options

As a virtual pin cab builder, you have several good options available. The right option is a matter of taste and budget.

Natural wood style. This isn't common, but some people choose to make their cabs look like a piece of fine furniture or cabinetry, to better fit into a home environment. If you want this kind of look, you can use a wood stain or a natural clear finish with cabinet-grade plywood. You can even buy pre-finished plywood to skip the staining step.

Single-color painting. This is another simple, understated look that some people use to make their machine relatively inconspicuous for the home environment (as inconspicuous as a six-foot-tall, five-foot-long, three hundred pound wood box can be, anyway). The most common single-color paint job is solid black, since that tends to disappear into the background nicely.

Stencil graphics. To a lot of people, the electromechanical era (1950s through 1970s) is the Golden Age of pinball, and tables from that era define what a pinball machine is supposed to look like. To be sure, the EM era's graphical style is unmistakably distinctive, and it's iconic of pinball in popular culture. The stencil graphic style that these machines used is also something that you can reproduce on your own, at low cost and without any special equipment. You just need to make a stencil mask out of cardboard and masking tape, and then apply spray paint in as many colors as desired.

Full-color decals. Many pin cab builders want to reproduce the look of machines from the modern era (1990s and onwards). These machines use elaborate designs printed in full color at high resolution. The real machines from the 1990s used high-res screen printing; newer machines almost all use plastic decals to achieve the

same look. Happily, professional custom decal printing is readily available for one-off print jobs, and is even relative affordable. This isn't something you can do at home with DIY equipment, since it requires special industrial printers, but there are lots of print shops that have the equipment and can do the job for a reasonable fee. And since the printing is done on what's essentially a giant industrial version of an ink-jet printer, you can print virtually any custom design by preparing the graphics with a PC photo editor program.

Using decals

Most pin cab builders these days opt for decals, since they allow for such unlimited creativity in the artwork.

First-time cabinet builders are sometimes skeptical about decals, thinking that they'll look like cheap stickers. It might reassure you to know that most of the newer real machines now use decals for their artwork, using the same materials that a good print shop would use for your cab decals. If you can find a newer Stern machine to look at, you can get a first-hand look at what kind of finish you can expect. When printed on quality stock and applied properly, you can achieve a finish that's pretty close to the screen printing used in the 1990s machines. Decal printing is actually superior in some ways; you get a wider color gamut and finer dot pitch, and the plastic finish is more resistant to light scratches.

Surface preparation for decals

You should check with your decal vendor for advice about surface preparation. I'd always give your vendor's advice priority over any generic advice you see on the forums or in build guides like this one. Different vendors use different film stocks, and what works best for one type might not be ideal for others.

With that in mind, I'll give you my own generic advice, based on working with a couple of different decal sources.

The first question is whether or not to paint before applying decals. I say yes, mostly because I want to make sure that any exposed wood areas around the edges of the decals match the decal background color, to hide the transition. Paint can affect how the decals adhere, so if your vendor says you should or shouldn't paint, I'd follow their advice; but if they say you can go either way, I'd paint.

Note that paint is probably required if you use a grain filler (which we'll come to shortly). Grain fillers don't adhere strongly on their own - they have to be sealed with paint or lacquer.

If your cabinet is built with plywood, the second question is how to prepare the surface, apart from the optional painting. No additional prep is necessary for MDF or MDO plywood, since the factory finish is paper-smooth. With regular plywood, though, the veneer has visible wood grain. Vinyl decals adhere so tightly that the wood grain will be visible through the decals if you don't take some additional steps. Wood grain showing through the decals isn't really a problem, but most of the commercial machines have a smooth finish, so I prefer to minimize it for a more "factory" look.

I think the only way to achieve a grain-free finish is to use a wood grain filler product before painting. My experience is that sanding alone won't completely eliminate the grain texture, no matter how much sanding you do or how fine the grit, because sanding won't eliminate the pores in the wood grain. The pores are what make the grain show through paint and decals, because they absorb the paint unevenly. Wood grain fillers are specifically designed to fix this by plugging up exposed pores at the

surface, so that paint is absorbed more uniformly.

Some pin cab people have reported good results using paint alone. I've had less than perfect results that way myself, and all of the kitchen cabinet pros seem to agree that grain fillers are a must if you want a piano finish, but maybe it's a matter of finding the right paint. It would certainly save a lot of labor to skip the filler step. Some paints, particularly primers, do make claims that they can serve as a surface defect filler.

If you do decide to use a grain filler, you can find lots of how-to videos about product selection and application procedures on the Web. Most of the videos I've found are focused on painting kitchen cabinets, so that's a good search term to use, but the same techniques work on pin cabs. If you want to skip the video research, here's a procedure that has worked well for me. It's time-consuming, but it doesn't require any special expertise or equipment.

- Sand the bare wood until smooth to the touch, with a power sander and 320 grit sandpaper
- Apply grain filler. I've had good results with Aqua Coat White Cabinet Grain Filler. Many people say that drywall compound (any brand) works just as well. If you use a grain filler, follow the manufacturer's instructions, but I think the basic procedure is the same for any of these products:
 - Work in small sections of a couple of square feet at a time, so that the filler doesn't dry out before you finish working the area
 - Apply a big glob of filler
 - Using a squeegee or gloved hands, work the filler back and forth **across** the grain, pushing it down into the grain
 - Immediately scrape off the excess using a stiff plastic squeegee or old credit card, scraping **with the grain**; scrape off as much as you can, leaving just what's trapped in the grain
 - Continue working in sections until you finish the whole panel
 - Let it dry (for Aqua Coat, 45-60 minutes)
 - Hand-sand **very lightly** with a fine-grit sanding sponge (e.g., 3M SandBlaster 320 grit) until the surface feels smooth. (This produces a lot of very fine dust. Wear a good respirator, and do it someplace where you don't mind getting dust everywhere.)
 - Repeat the steps above for a second coat; this one should take much less filler, since most of the open grain will already be filled from the first coat
 - Repeat for a third coat, or as many more coats as necessary until the grain is well concealed
 - Let the last coat dry overnight, and lightly hand-sand smooth
- Clean off all surface dust by wiping and/or vacuuming (avoid moisture here, since most grain fillers are water-soluble - damp cleaning might undo all of your work so far)
- Apply two coats of a filler primer paint. I've had better results spraying than brushing (in my case, using spray paint in a can: I used Rust-Oleum Filler Primer on my last build, and that worked pretty well). My experience with brushes and rollers is that both can leave a texture that shows through the decals.
- Optionally, sand very lightly with fine or ultra-fine sandpaper or a sanding sponge (400 grit or higher). Think of this more as polishing than sanding.

You're just trying to smooth out dust bumps and air bubbles. I've had better luck **not** using a sanding block for this - it seems too easy to make scratches that way. Instead, I just use a small piece of sandpaper, hand-held. Work in small areas and check progress frequently by touch.

- Wipe clean again with a slightly damp cloth. Apply two or more top coats of paint that matches the background color of your decals. After each coat fully dries, you can do another extremely light polishing/sanding pass. It's really important to let the paint dry long enough to fully cure and harden before sanding, otherwise the paint can ball up and come off in little chunks, defeating the whole purpose. Check the instructions on the paint can for full drying time.

A lot of people like to do the sanding between coats as wet sanding (dipping the sandpaper in water, or water with a tiny amount of mild soap). Only do this with oil-based paints. Wet sanding can create an even smoother finish and helps avoid scratches.

Applying decals

Decal application is scary the first time you do it, especially since the decals are expensive, and there are at least a few horror stories on the forums about how difficult decals are to work with. But it's one of those things where you don't need special magical skills. If you follow the right procedure, you should be able to get good results reliably.

There are two basic techniques: the "wet" and "dry" methods. This is one of those topics that inspires an almost religious fervor in a lot of people: proponents of the wet method will tell you that you'd have to be crazy to even think about the dry method, and advocates of the dry method will say the same thing if you're contemplating the wet way.

The "wet" method involves spraying the cabinet surface and the back of the decal with a soapy solution just before application. Some decal film stock requires this as a way to release trapped air bubbles, but newer, higher-tech decal materials are designed with tiny pores that release air bubbles on their own, eliminating the hard requirement to use the wet method. Even so, some people still like the wet method for a whole separate reason, which is that it keeps the decal from attaching too strongly at first, so that you can slide the decal around to fix any initial alignment errors.

The "dry" method simply applies the decal directly to the clean, dry surface. Newer films don't need any help releasing small air bubbles, so there's no need for soapy sprays. The decal adheres strongly right away with this method, so you don't get to slide it around to play with alignment - but you shouldn't have to do that if you use the right procedure, because you'll get it aligned beforehand.

You can find Youtube videos for both methods. This is a good subject to preview on video so that you can get a little mental practice before attempting it. Search for "pinball decal dry method", for example.

As with surface preparation, I'd always take your vendor's advice on application method over anything generic that you see in the forums or from me. Some media might simply require the wet method, because of the air bubble issue that affects some film types. On the other hand, some decals might not be able to tolerate too much added moisture.

Personally, I prefer the dry method. It's the one that my decal vendors have all recommended, and it seems simpler and cleaner to me. I can understand the appeal of the do-over potential of the wet method, but at the same time, it seems prone to

a little less accuracy exactly because of the slipperiness.

The key to making the dry method work is to lock in the alignment before you expose the adhesive. Here's the procedure I use:

- Before starting, have a felt squeegee ready. Some people like the really large ones that can go across the whole width of the decal, but a small one works too - the one I use is 4" wide.
- Wipe down the cabinet surface **and** the back of the decal (don't peel it - I'm just talking about wiping down the back of the backing paper). A clean soft cloth with a little bit of rubbing alcohol works well for this. The point of cleaning the back of the decal is that you're going to have to lay it across the cabinet surface, so any dust on the back of the decal can transfer to the cabinet surface, undoing your careful pre-cleaning.
- Initially, leave the backing paper in place, and get the decal aligned exactly how you want it on the target surface.
- Once it's lined up correctly, temporarily fix it in place at one end, using clamps or masking tape. Make sure to tape it or clamp it securely, so that this end of the decal can't move.
- Lift the other (unpinned) end, making sure that the fixed end of the decal stays stuck in place. Peel the backing away from the free edge for about two or three inches.
- Cut off that first two-to-three inches of backing with scissors or a utility knife. Leave the rest of the backing in place.
- Working from the pinned end towards the free end, smooth the decal flat against the surface again. The exposed section will now adhere to the surface. The decal is now serving as its own anchor at this edge.
- Remove the clamps or masking tape.
- Working from the end that's already stuck, lift the decal enough that you can start peeling off the remaining backing.
- Peel off a little bit of backing at a time, and smooth the decal onto the surface as you go, using your felt squeegee. Just do a couple of inches at a time. If your squeegee is the 4" type, sweep it back and forth across the width of the newly stuck section. If it's the super-wide type, just keep moving it down the length of the decal.

It's really important to keep the unstuck part of the decal straight throughout this, so that it doesn't form any wrinkles and doesn't stick to itself anywhere. Maintain light tension on the free end.

Trimming edges

Most print shops will print the decals slightly larger than the final size you want to install, usually about an extra inch on each side. This is intentional; it's to give you a little room for error in the final alignment.

The standard procedure is to align the decals, affix them, then go around the edges with an X-Acto knife to trim the decals to be exactly flush with the edges. This is surprisingly easy; you just let the edge of the wood guide the knife. As long as the knife is sharp, it should make a perfect cut exactly at the edge.

Cutting out holes

When you design and apply the decals, you should simply let them cover the holes in

the cabinet for the flipper buttons, front panel buttons, and coin door cutout. After installing, use an X-Acto knife to trace around the edge of each opening. Cut from the **outside**, and let the edge of the opening guide the knife - the same procedure used to trim excess material around the edges.

Finding a printer

My decals were printed by Brad Bowman a/k/a Lucian045 on VP Universe (also reachable at bjbowman045@gmail.com). I highly recommend him. Brad is a serious virtual pinball enthusiast who also happens to run a professional sign printing shop. It's great to work with a printer who knows how pin cabs are set up, because that means he'll be able to picture what you have in mind for any special customizations. The decal stock that Brad uses is also just great: very easy to work with and very durable. I of course can't guarantee that Brad will still be offering print services by the time you read this, but you can always drop him a line to find out.

Other options include VirtuaPin and GameOnGrafix.com. Both offer custom decal printing. VirtuaPin specializes in pin cabs and I think they use similar print stock to what Brad Bowman uses. GameOnGrafix is more oriented towards home-brew video game cabs, but they also provide a template for pinball cabinets, and anyway it's basically the same sort of decal for either type of machine.

You can also try any shop that does commercial sign printing. This is a common commercial service, so you can probably find local vendors in your area, especially if you live near a major city. The type of adhesive plastic material used for pin cab details is also commonly used for commercial signage.

Artwork requirements

Most print shops will expect you to provide your artwork in an electronic format, such as JPEG or TIFF. Check with your vendor for their requirements and recommendations. You should be able to use just about any photo editor or painting program on your PC to create the graphics and convert them into the vendor's preferred format.

Decal printing is essentially the same as printing on a home ink-jet printer. The only real difference is that the decal prints are physically a lot larger. So keep in mind that the pixels you see on the computer screen will be spread out over a much larger area when printed. Images that look smooth and sharp on-screen might be fuzzy with jagged edges when blown up to pinball decal size. To look good at full size, the final image will need a pixel resolution of about 300 dots per inch (dpi) when printed. The side panels of a full-sized pinball machine are about 50" x 24", so if you want to fill that space at 300dpi, you'll need the source image to be about 15,000 pixels by 7,200 pixels - about 100 Megapixels.

Creating your artwork

There are three main options for creating your artwork.

Design it yourself. If you're feeling creative and you're good with a graphics editor like Photoshop or Illustrator, you can design your own original artwork.

Opting for a completely original design gives you the freedom to come up with whatever look appeals to you. But starting with a blank page is also pretty intimidating. Here are some ideas for where to begin:

- If you want to create something in the style of the real machines, start by choosing an era. Go to IPDB and browse through pictures of machines from the era, to get a sense of the prevailing graphic style. If a particular machine's

design strikes you as particularly appealing, use that as your starting point.

- Choose a name for your machine. That will automatically plant some ideas about its theme. A lot of pin cab builders name their machines after their favorite movie, TV show, or comic book character, following the long tradition in the real machines of using licensed themes.
- A popular motif is to focus on the virtualness of the machine and/or its ability to run many different games: "Multiball", "Megapin", "Pinball Holodeck", "Pinball Matrix", etc.
- Another way to emphasize the multi-game aspect of a virtual cab is to use a collage of prominent artwork elements from your favorite real pinball machines, such as Rudy from *Funhouse* and the Addams family characters from *The Addams Family*.
- There's a lot of public-domain (copyright-free) artwork on the Web that you can use as a starting point. For example, if you like space themes, check the NASA, JPL, and Hubble Space Telescope Web sites for some very pretty, high-resolution astronomy images that are free to use. I used a Hubble photo of the Carina nebula as the backdrop for my own cab side art. (Do be sure that any images you take from the Web are truly public-domain or licensed for free use. Reputable print shops won't accept artwork that you don't have the proper rights for.)



Commission original custom art. If you're not interested in creating your own artwork, but you still want something original, you can find an artist to create something custom for you. For example, stuzza on vpforums creates original art for forum members, for a fee. A stuzza design is generally a pastiche of pop culture clip art based on a theme you provide. See the long-running thread "Cabinet Artwork I have created" for his contact information and examples of his work. VirtuaPin also offers custom graphic design services for a fee.

Use a pre-made design. Stuzza on vpforums has also released a number of free designs that you can download and use without a commission fee. See the "Cabinet Artwork" thread mentioned above for links. I've also come across occasional pin cab artwork elsewhere on the Web; try an image search for "pinball cabinet side art".

Reproduce artwork from a real pinball. Some cab builders opt to use the original artwork from their favorite real machine. Be aware that the graphics from virtually all historical commercial machines are still under copyright, so a reputable print shop won't accept an order that reproduces a real machine's artwork without proper clearance from the rights holders, which almost always requires paying a license fee. VirtuaPin sells authorized reproductions of the original art for several popular classic pinball titles. You can also find ready-to-use decal sets with reproduction artwork

from many more titles from pinball supply vendors - search for "pinball cabinet decals".

Backbox warning label

Most commercial machines display a big block of warning text on the back of the backbox, warning operators to bolt the backbox properly and fold it down for transport. The warnings were there for the usual legal liability reasons, so if you're just building a cab for your own use at home, you can leave the area blank. But some cab builders might like to include the warnings for the sake of meticulous re-creation of the originals. See Chapter 43, Extras - Backbox warning label for a picture of the typical text.

23. Cabinet Hardware Installation

This section covers the hardware trim on the main cabinet, with details on what all the hardware does and how to install it.

We assume that you've already built the basic plywood cabinet as described in Chapter 21, Cabinet Body, and that you've already painted it and/or applied decals, as described in Chapter 22, Cabinet Art. It's best to finish the artwork before installing any hardware, since some the hardware will get in the way of painting or applying decals once installed.

The sections below are arranged in our recommended order of installation. The order matters in some cases because of dependencies among parts. For example, you should install the side rails before installing the lockbar, since the lockbar's final alignment depends on the side rails being there, and you have to install the glass channels before the side rails because the channels act as spacers and supports for the rails. We've tried to present things in the easiest overall order, to get alignments right the first time and to minimize backtracking.

What to buy

We'll describe the parts needed at each stage as we go, but the full list of hardware parts can be found in Chapter 20, Cabinet Parts List.

Leg brackets

You should attach the leg brackets early in the cabinet build, during the basic wood assembly. The brackets are meant to be permanently installed; they can be left in place with the legs on or off the machine.

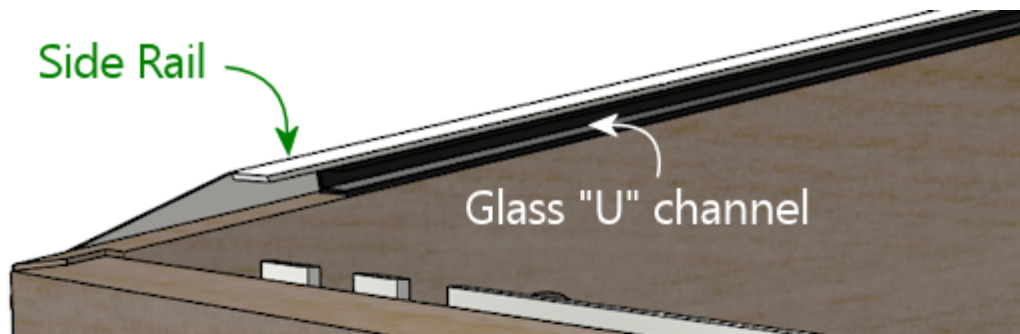
If you followed our plans in Chapter 21, Cabinet Body, you should have already installed the leg bolt brackets on the inside of the cabinet. If you haven't done this already, see the section on Chapter 21, Leg brackets in that chapter.

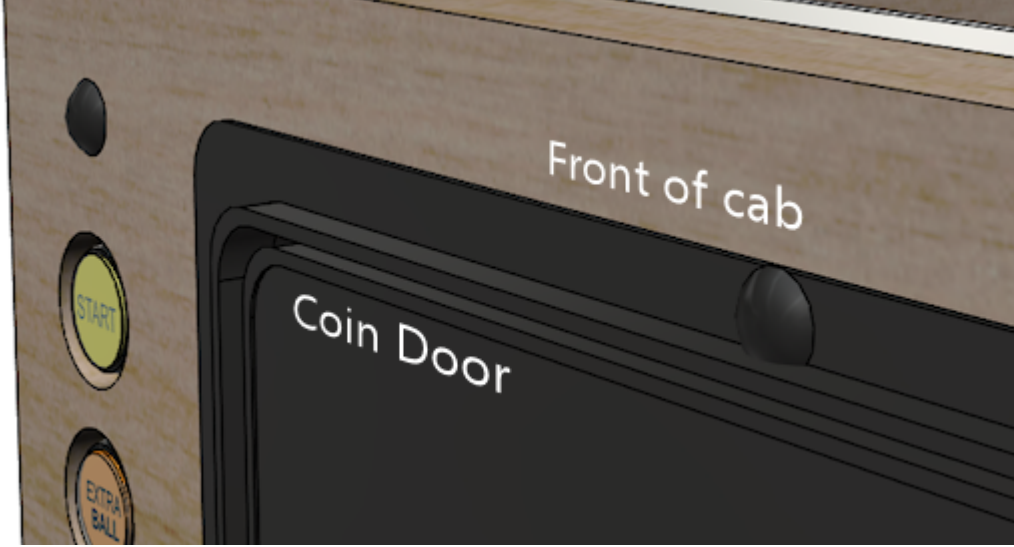
You probably won't want to attach the legs themselves early on, since you'll probably want the cab sitting on the floor or workbench while you install the PC, TVs, and other internal components.

Note that we won't get to the legs themselves until near the end of this chapter, as you'll probably want to keep the cab on the floor or on your workbench while you're installing the PC, TVs, and other internal components.

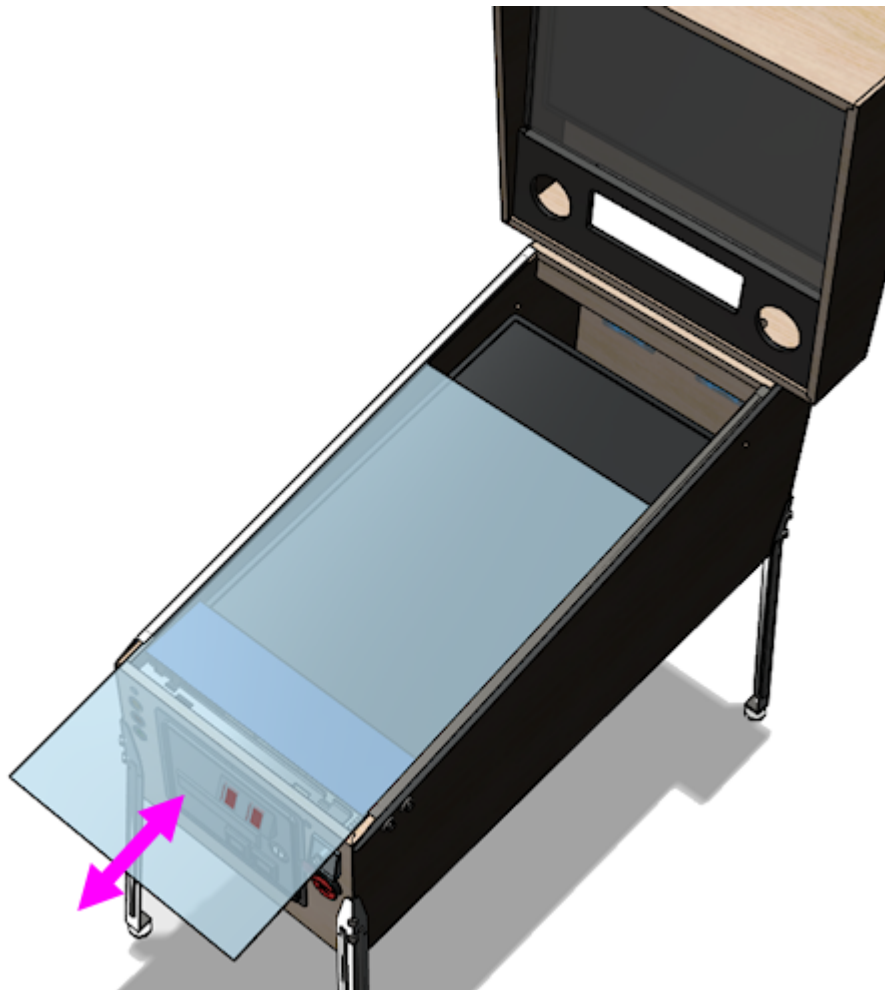
Glass channels

These are plastic pieces that go under the side rails, to hold the top glass cover in place.



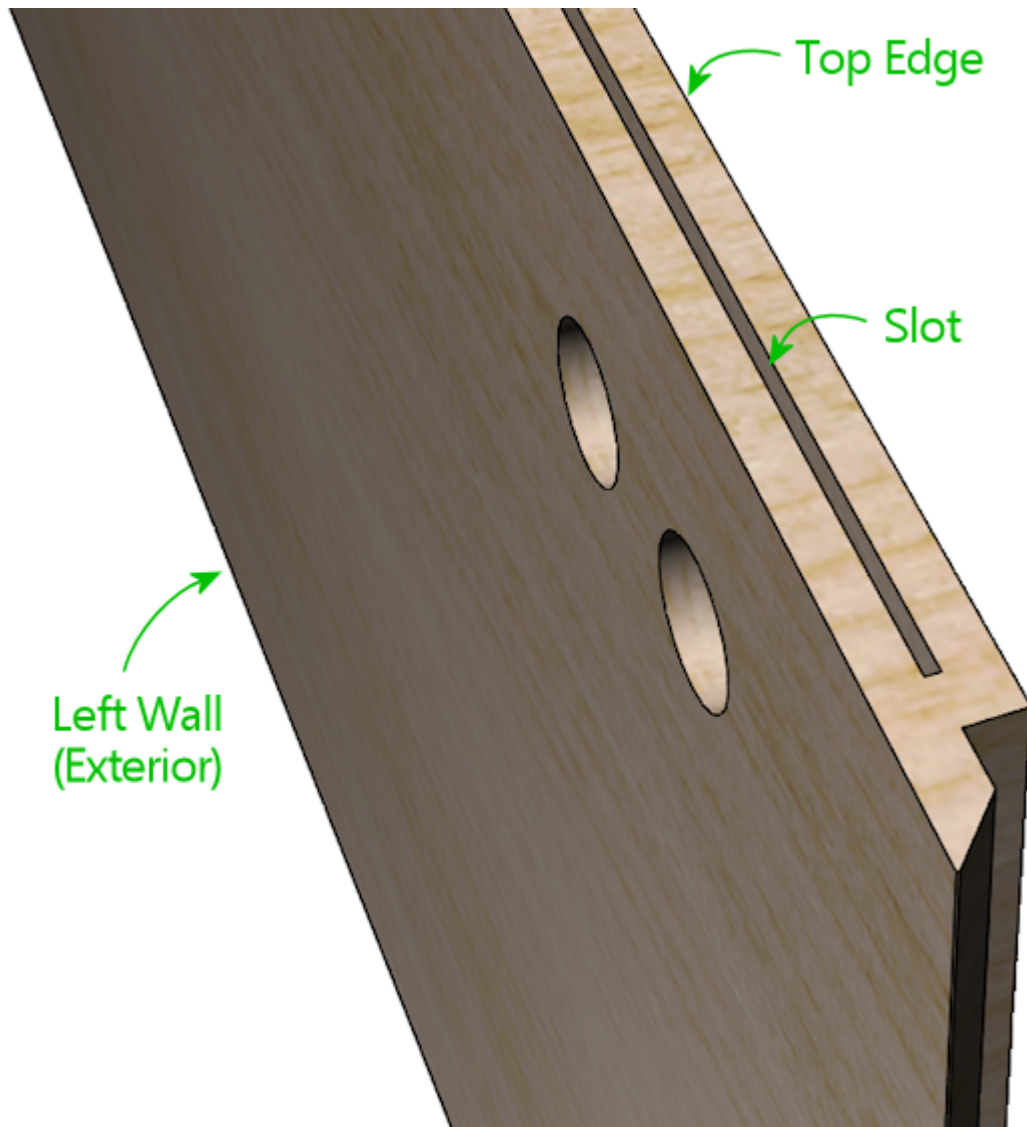


The glass channels aren't visible, so they're not "trim" in the cosmetic sense, but they have two important functional roles. The first is that they let you slide the top glass in and out of the machine at any time, without tools. To remove the glass, you just unlatch and remove the lockbar, then slide the glass out the front. To put the glass back, slide it in the front, and pop the lockbar into place.



The second important functional role of the glass channels is to act as vertical spacers and supports for the metal side rails. All of the side rails made since the 1970s or so are designed to be paired with the glass channels. The geometry of the rails simply takes it for granted that the plastic channels will be there - if they're not, there will be a big gap under the rails, which would make them too weak.

To install the glass channels, the first step is to route a groove along the center top of the side wall. (If you followed our construction plans in Chapter 21, Cabinet Body, you should already have done this.) There's a special router bit for this job, called a slot cutting bit. For this particular slot, you need a slot cutter bit with a $\frac{3}{32}$ " slot width and a depth of $\frac{3}{8}$ " or greater. I used Freud part number 63-106; equivalent bits are available from other brands.



Alternative to routing the slot

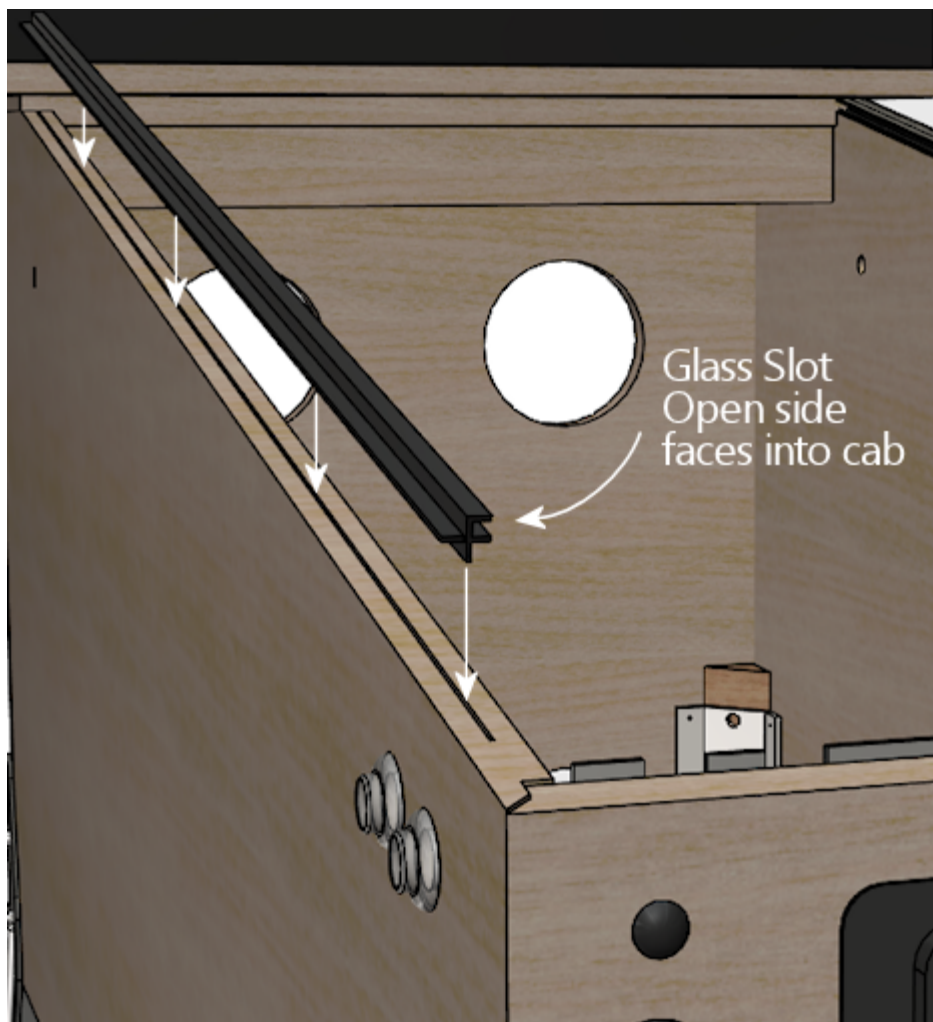
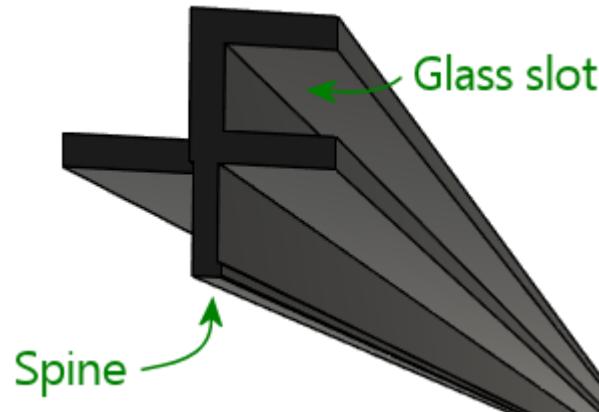
Some virtual cab builders forego routing the slot, because it seems too difficult, or just because they don't want to buy a special tool that they'll only ever use once. But they still want to install the channels. How do you make the spine fit without the slot? You can't, but you *can* chop off the spine to take it out of the picture. The glass channels are a fairly strong plastic, but they are just plastic, so it's possible to remove the spine with a sharp utility knife, a Dremel tool, or something similar. With the spine removed, you can install the channel with staples or glue. You don't have to go overboard with a super-strong attachment, since the metal rail will sit directly on top of the channel, and that by itself will largely keep it from going anyway.

Personally, I don't recommend this approach. I'd install it the "right" way with the slot, since it makes a tidier installation and doesn't risk damaging the plastic piece. Cutting the slot probably seems intimidating if you haven't done that sort of thing before, but the special bit makes it really easy.

Installing the channel

Once the slot is routed, install the plastic channel simply by pressing the "spine" that runs along the bottom of the channel into the plywood slot. **Don't** use any glue or fasteners; the spine will be held in place by friction, and if that's not enough, the side rail on top of it will prevent it from going anywhere.

You should expect the spine to be a tight fit in the slot. You'll need to press it in fairly firmly. Don't force it too aggressively, though - you don't want to split the plywood. If the channel is way too tight, you can always go over the channel with the router bit again to widen it out slightly.



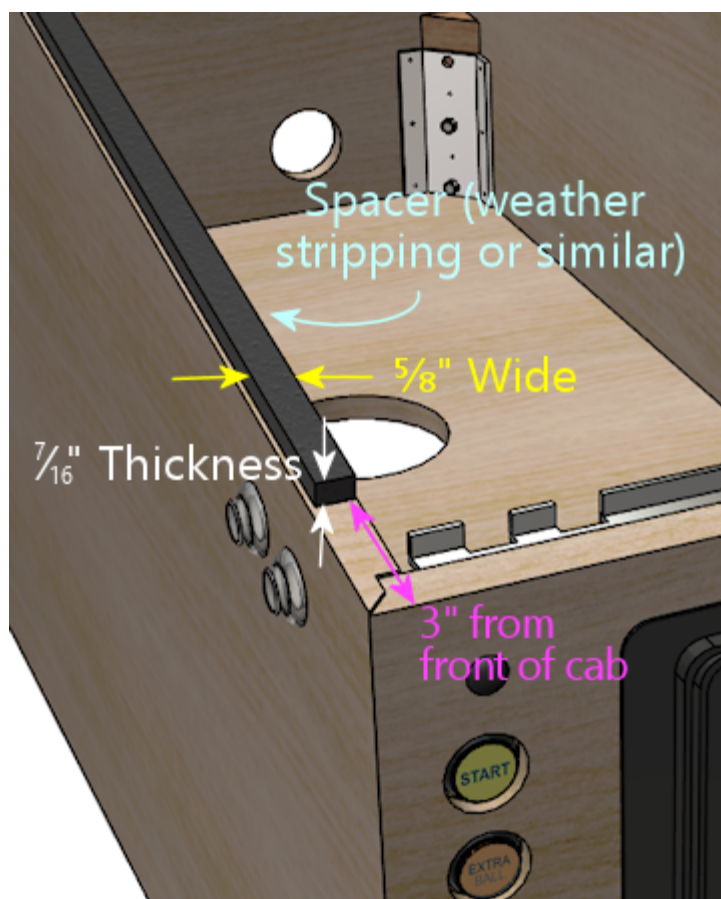
Alternatives to the glass channels

What if you don't want to include the glass channels? Not all virtual pin cab builders do, either because they're not using the cover glass at all, or because they want to install it some other way.

My own recommendation is to use cover glass and use the standard glass channels. That'll look the most authentic and it'll be easier to set up than something improvised. But if you're intent on another approach, you'll still need *something* to serve as spacers under the rails.

If all you need is to fill the space, and you don't care about sliding the glass in and out, you can use any material of the right size. Something like foam tape or rubber weather-stripping material would work. The target size for the spacer is about 7/16" thick, 5/8" wide, and 42" long. Affix your chosen spacer material with glue or double-sided foam tape, starting about 3" from the front of the cab.

Be sure to take the thickness of the adhesive into account when sizing the spacer - the **total** overall thickness (height) of the spacing material should be about 7/16", **including** any adhesive layer. If you're using foam tape that's 1/4" thick, for example, the spacer material would only need to be about 3/16" thick.



*Improvised spacer material under the side rail. **Only** use this if you're **not** using the normal glass channels! The point is to substitute something that provides the same vertical spacing and support as the glass channels if you're **not** using the actual glass channels.*

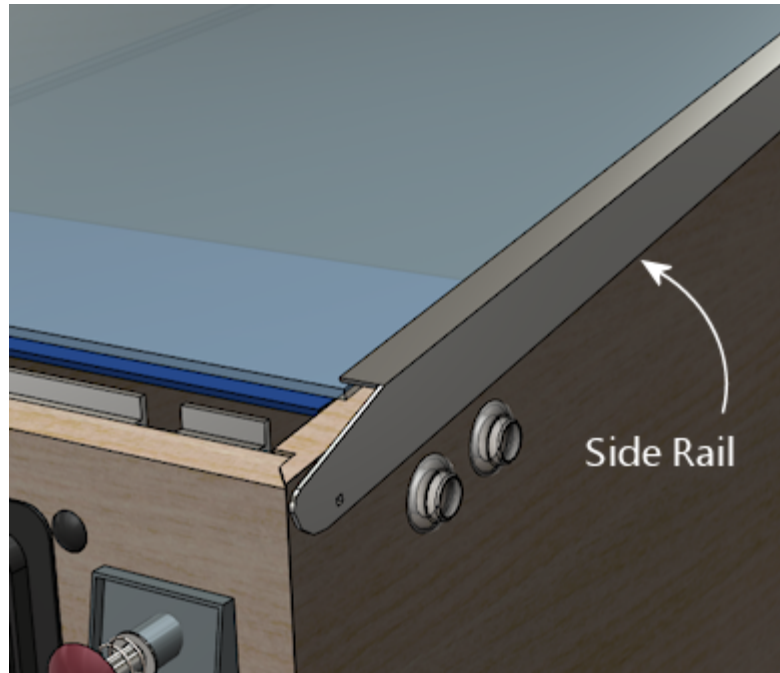
If you're using cover glass and want to be able to slide it in and out, I don't think there's any reason to look for alternatives to the standard plastic channels. They're the exact right thing for the job.

Side rails

These are trim pieces that go along the top side edges of the cabinet. On all modern machines, they're made of sheet metal, usually with a stainless steel finish.

(In the very early days of pinball, the side edges were trimmed with wood molding

instead of metal, which is the source of the term "wood rail pinballs" that's often used to refer to such machines. If you want to use something like that, it would require custom wood-working; there's no such thing as a "standard" wood side rail, and nothing of the sort that you can buy off-the-shelf, as they haven't made commercial wood-rail machines in many decades.)



*WPC-style side rail. Note that the rail is narrow enough along the side wall that it doesn't reach the flipper buttons. Older side rails (from the 1980s and before) extend further down the side and typically **do** cover the flipper buttons, requiring holes in the rails for the flipper buttons.*



The WPC-style rails

The design of the side rails varies by manufacturer and vintage. The illustration above shows the WPC style from the 1990s, which continues to be the standard for everything made since. It's the type I'd recommend if you're buying new parts from a pinball supply vendor. It's the easiest type to find, too, since it's the only type anyone has been using in new machines for the last 30 years or so.

If you're using side rails salvaged from an older (1980s or earlier) machine, they might look different. The big difference with the older designs is that they're usually much wider, extending down the sides far enough to cover the flipper buttons. Because they overlap the flipper buttons, the older rails usually have pre-drilled holes for the buttons - so you have to be careful to drill the cabinet button holes so that they line up with your rails. The WPC rails don't have any such alignment requirement because they don't overlap the flipper buttons. The only alignment you have to worry about for them is making sure that the buttons are entirely below the rails.

The WPC-style rails are symmetrical. They don't have "left" and "right" versions because the same rail can be used on either side. Simply flip the rail over to switch sides. The older rails with pre-drilled flipper button holes can't do that trick, so they

come in left/right sets.

Fasteners

Three fasteners are required for each side rail:

- At the front, an #8-32 x 1-1/4" carriage bolt, #8-32 ESN lock nut, and #8 washer
- Along the sides, double-sided foam tape, about 3/4" wide by .03" thick
- At the back, a spiral nail (Williams part numbers FA-701, 20-6505-L, or 20-6505-K), or a plain nail or wood screw (see notes below)

Foam tape: Side rails sometimes come with foam tape already installed. If yours didn't, you can buy the tape separately from a pinball vendor (search for "side rail tape"), or use a generic double-sided foam tape. You need tape that's about 3/4" wide and about .03" (0.75mm) thick. (The original Williams specs called for .032" thickness, but you don't have to match that perfectly.) Each rail takes 40" of tape, for 80" total.

Rear fasteners: The original machines used spiral nails to fasten the rails at the back. Spiral nails have flat, smooth heads like nails, and spiral ridges like screws. You pound them in with a hammer like regular nails. The spiral ridges give them a lot more grip than regular nails, to prevent loosening from vibration.



Spiral nails, 0.1" diameter x 1" length, Williams/Bally part 20-6505-L. These are traditionally used to fasten the side rails at the back.

I actually don't much like the spiral nails, because they're quite difficult to remove, which causes a lot of grief to collectors restoring old machines. You might wonder (I sure did) why you can't use the same carriage bolts as at the front. The answer is that the shelf gets in the way on the inside - you can't access the fastener from the inside to attach a nut. So you need a fastener that screws in or pounds in purely from the outside, without the need to attach anything on the inside. Spiral nails fit the bill, but so would a plain nail or wood screw. The next question, then, is why not just use regular wood screws? You could simply unscrew them if you ever wanted to remove them. Part of the rationale at the factory might be tamper resistance, which you probably don't care about for home use, but the other reason is that whatever fastener you use has to fit into the tiny gap between the side rails and the backbox hinge arms. Most nails (including the spiral nails) meet this need. Most regular screws don't; the heads are too thick. You need something that's 1mm or less. For reference, the spiral nail heads are about 0.8mm thick.

Here are the thinnest options I've found in a wood screw:

- "Pancake-head" roofing screws, such as Bolt Depot part number 27072 (#10 x 1" combination Phillips/square drive, pancake-head, type 17, zinc-plated steel)

- Grabber BP32Z screws (#8 x 1" Phillips drive, modified truss head, fine thread)

The heads on both of those are quite thin as screws go, but they're still too thick for our purposes. The heads on the Bolt Depot roofing screws are about 1.6mm thick, and the Grabbers are about 2mm thick. That's thicker than our maximum of 1mm. One solution is to modify these screws, by using a metal file or grinding wheel to shave some material off the tops until they're down to 1mm.



Grabber BP32Z screws, original condition on left, and with the head filed flatter on the right. The original head is about 2mm thick; I was able to get this one down to about 1.2mm, which was thin enough to fit under my backbox hinges.



Top view of the BP32Z screws, original on left, filed down on right. I filed it until there was no evidence of the domed part in the center, leaving the head almost as flat as a nail head. There's still enough of the Phillips indents to work with.

Alternatively, you might be able to make a flat-head wood screw work, if you can countersink the conical part of the head far enough to make the top more or less flush with the rail.

You could also use a plain old nail, if you don't care about eventual removal and you don't want the bother of finding the unusual spiral nails. A regular nail might work itself loose over time more easily than a screw or a spiral nail, but that's probably not a huge concern for a home-use-only machine.

If you find another option that works, please let me know and I'll mention it here.

Install the glass guides first

Before installing the side rails, install the plastic channels that hold the top glass (see Glass channels above). If you're not planning to use the standard plastic channels, you should install some kind of spacers of roughly the same size. Part of the function of the channels is to act as vertical supports for side rails, so you need something to fill that space if you're not using the plastic guide channels.

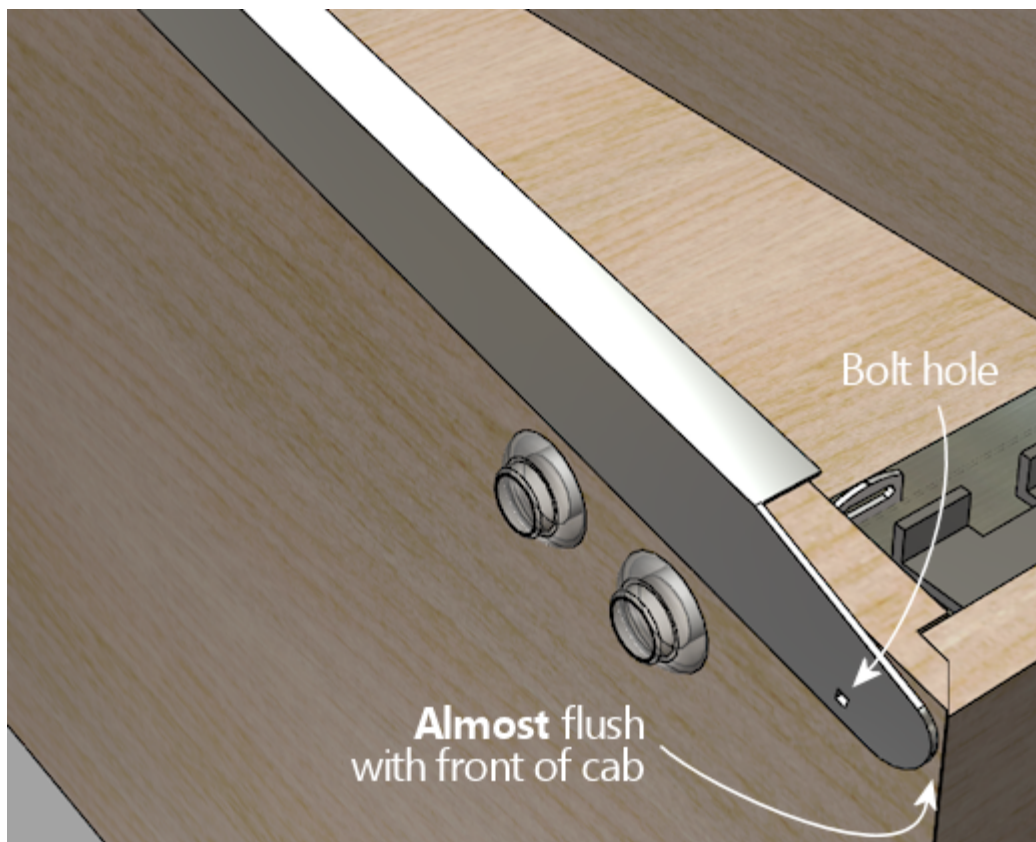
Installing the side rails

Start by sticking a strip of the double-sided tape to the inside of each rail. The tape should cover most of the length of the "L" shaped part of each rail, out to about an inch from each end. Don't put any tape on the tapered end sections.

The tape goes along the inside side surface - the surface that will face the side wall of the cabinet. Only peel off the backing on the side facing the metal rail at this point. Leave the backing in place on the other side.



Next, do a "dry fit" to the cabinet, placing the rail in position, resting it on top of the glass channel. The front of the rail should be **almost** flush with the front wall of the cab. Leave just enough of a margin ($1/16"$ to $1/8"$) that it doesn't stick out at all, so that it won't snag anyone's clothing.



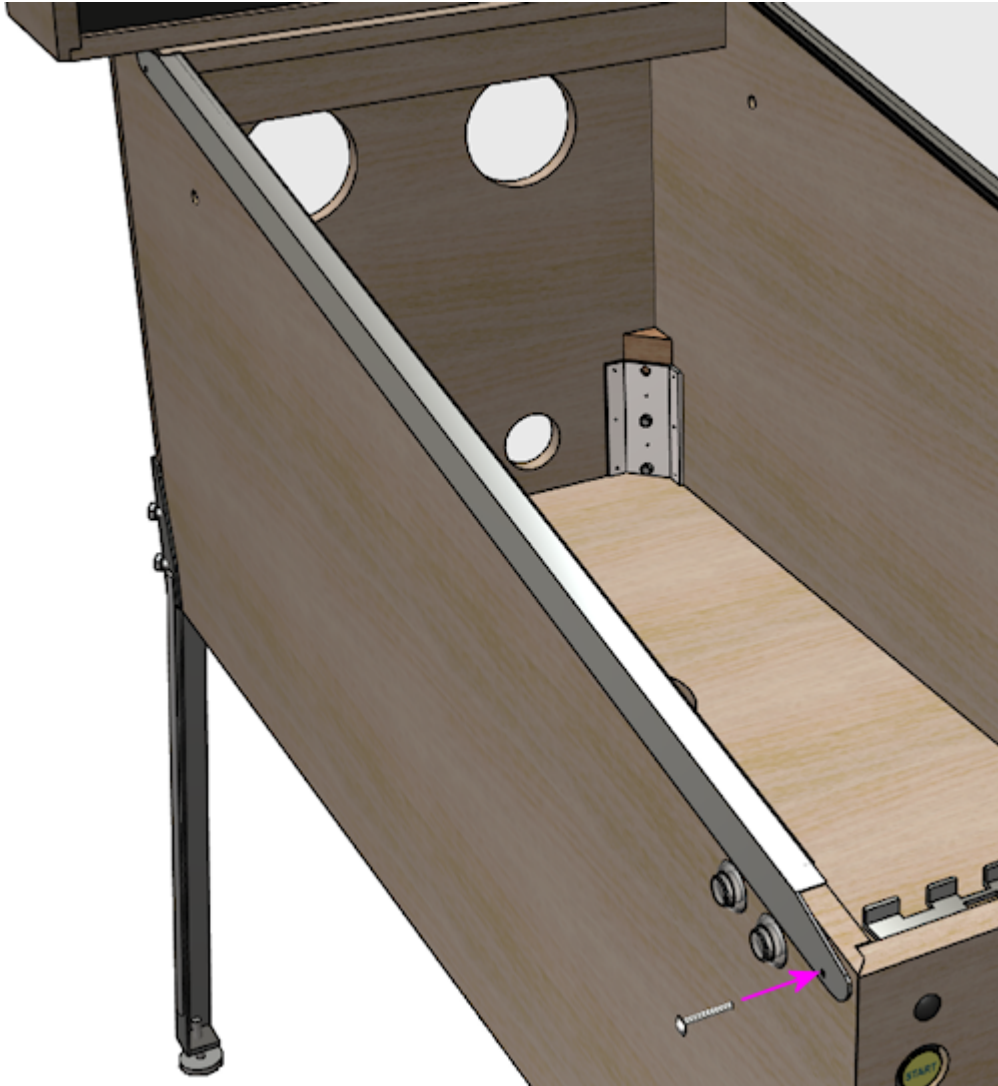
We're going to assume that you haven't drilled the holes for the side rail carriage bolts yet. This is the time to do that. Mark the cabinet positions where the front holes in the rails line up. Only drill for a carriage bolt at the front; at the back, you'll have to use a screw or nail, because a bolt would conflict with the shelf. Remove the rails and drill at the marked positions (straight through) with a $11/64"$ drill bit.

Warning: if you installed decals that cover this area, you might want to use an X-acto knife to cut an opening in the decal before drilling, so that the drill bit doesn't catch on or pull at the decal while you're drilling. This area will be covered by the side rail once that's installed, so the decal cut-out won't be visible.

You're now ready for final installation. There are two ways to proceed from here: you can stick the rail to the cabinet with the foam tape, or you can leave the tape unsticky for now and only use the front bolt. The first way - using the tape - is the way

it's meant to be done for permanent installation. The second way - without sticking the tape yet - is better if you're not ready to commit to the final setup yet (for example, if you might want to touch up the artwork later). The bolt will hold the rail in place well enough for testing and casual play, so there's no hurry to finalize the tape yet.

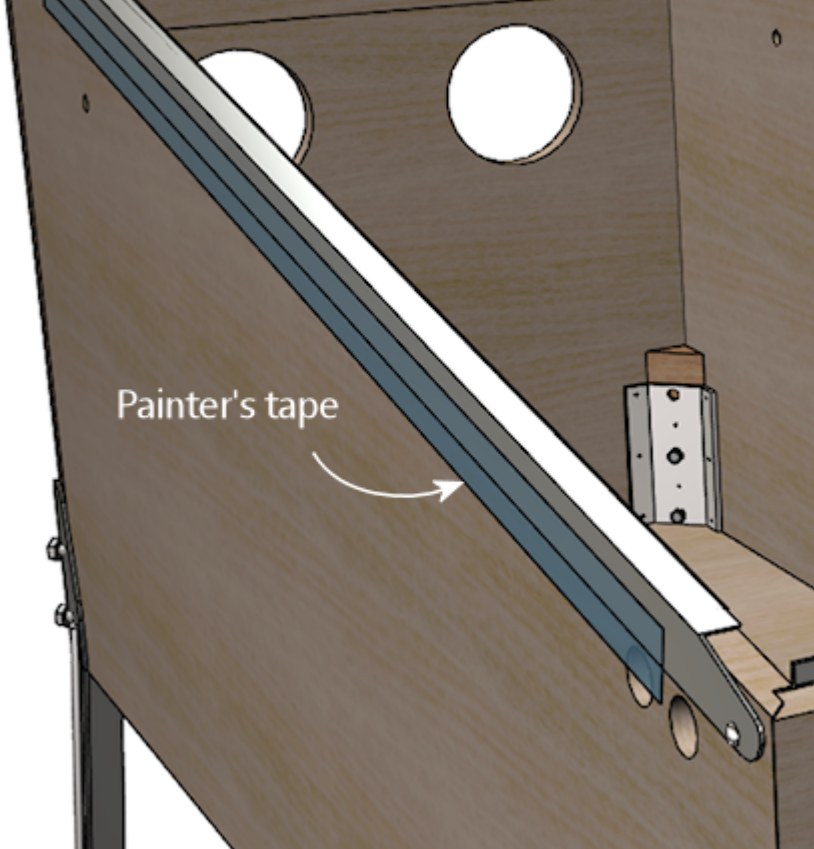
In either case, start by setting the rail in place again and lining it up with the carriage bolt hole you just drilled. Insert the bolt from the outside, and attach the nut on the inside. If you **don't** want to finalize on the tape yet, just tighten the nut, secure the back of the rail (with a spiral nail, plain nail, or wood screw, as mentioned earlier), and call it done for now.



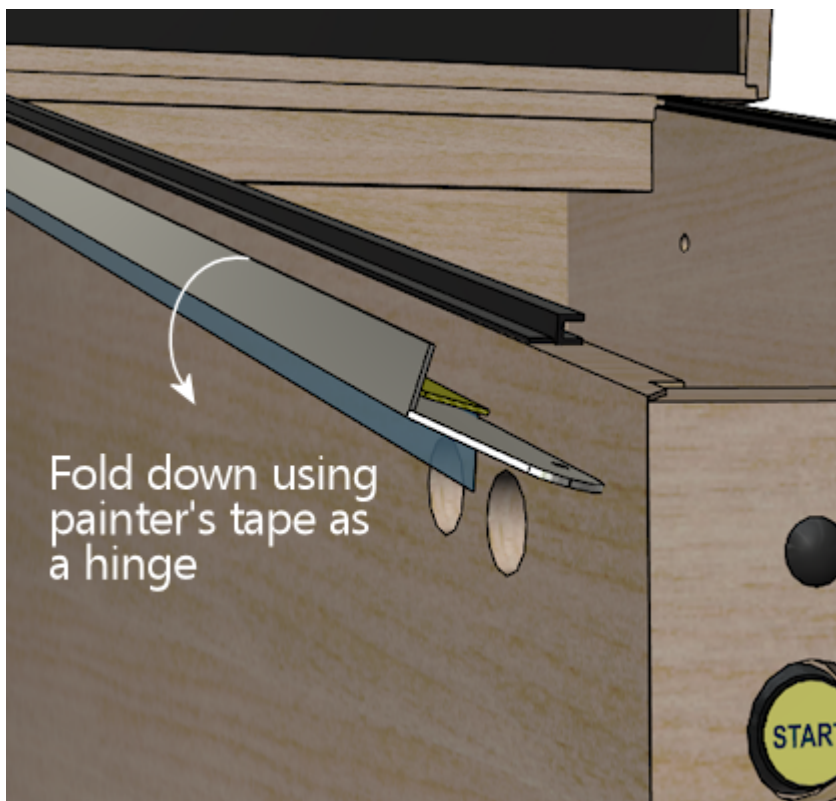
If you want to finalize the installation by attaching the double-sided tape, here's the recommended procedure:

- Start with the side rail bolted on as described above.
- Run a length of wide painter's tape along the bottom edge of the rail, down its whole length, centered on the bottom edge. In other words, half of the width of the tape should be sticking to the side rail itself, and the other half should be sticking to the adjoining side of the cabinet.



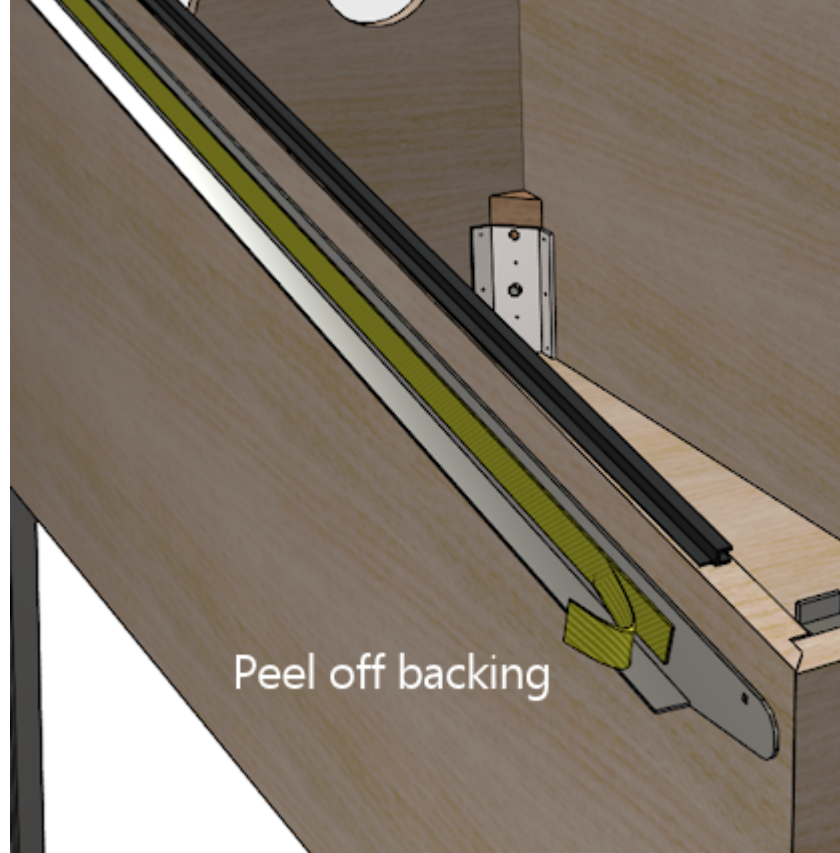


- Remove the bolt.
- Leaving the painter's tape in place, fold the side rail down, using the painter's tape as a "hinge". The rail should now be hanging upside down from the painter's tape, with its inside face (and the double-sticky tape) exposed.

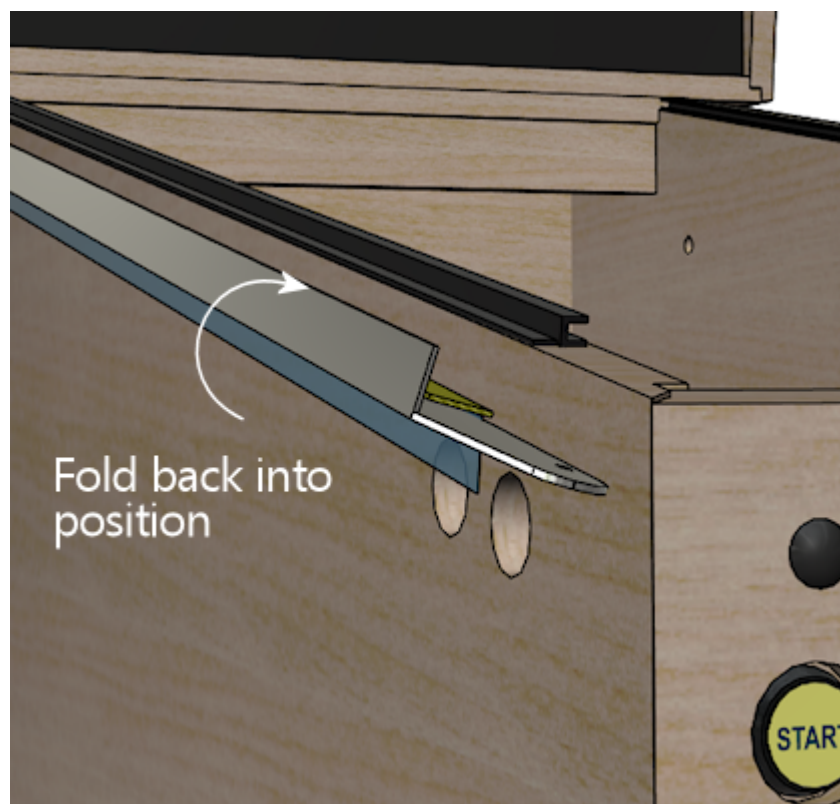


- Remove the adhesive backing from the double-sticky tape.





- Carefully fold the side rail back up and into position, using the painter's tape as a hinge again. This should precisely return it to the original position.



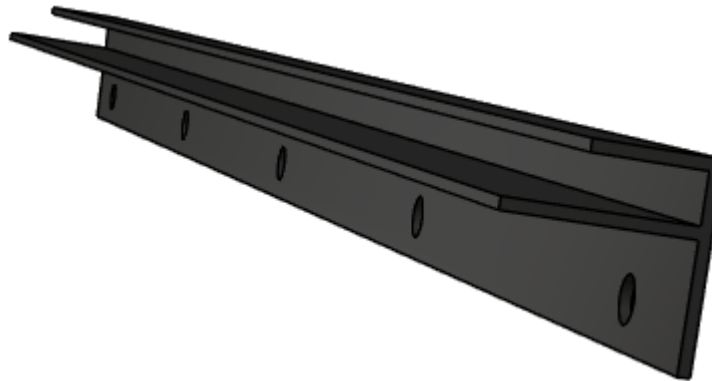
- Press along the side to adhere the double-sticky tape to the cab wall.
- Re-fasten the carriage bolt.
- Install the fastener at the back (spiral nail, plain nail, or wood screw, as mentioned earlier)

- Remove the painter's tape.

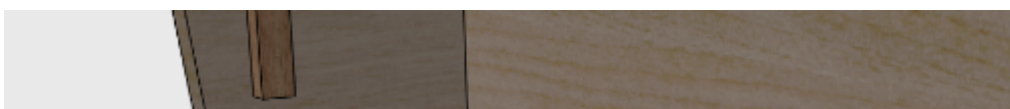
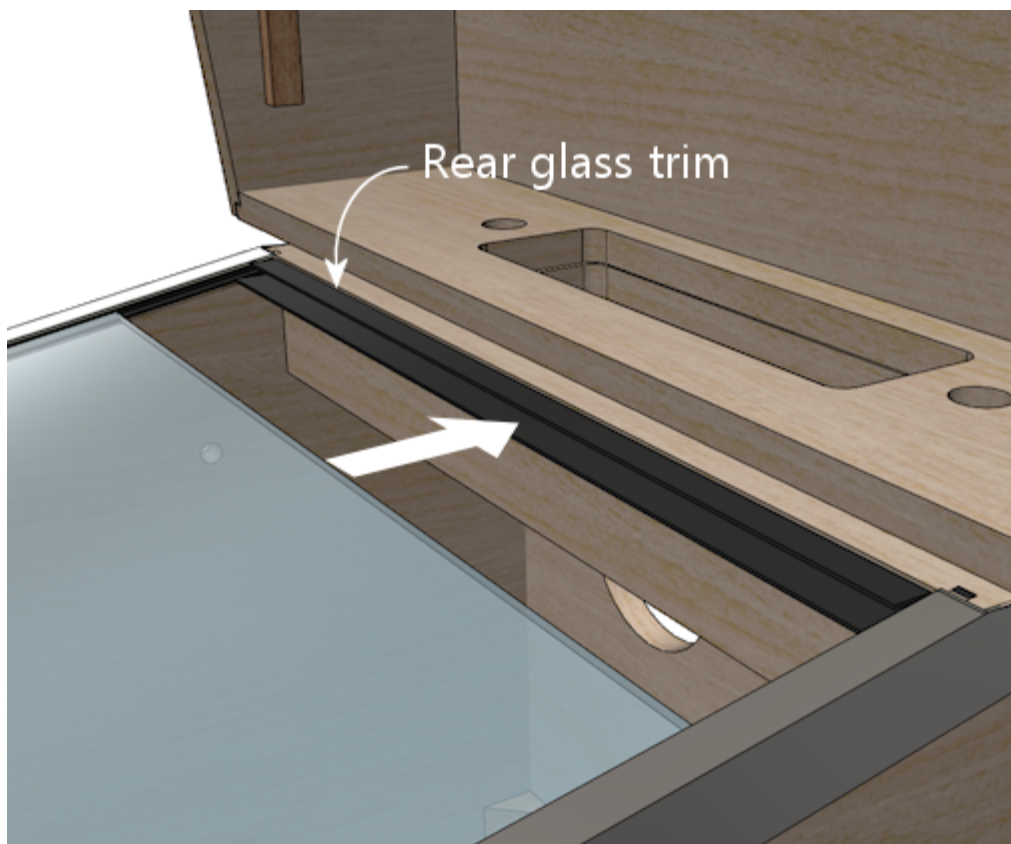
Rear glass trim

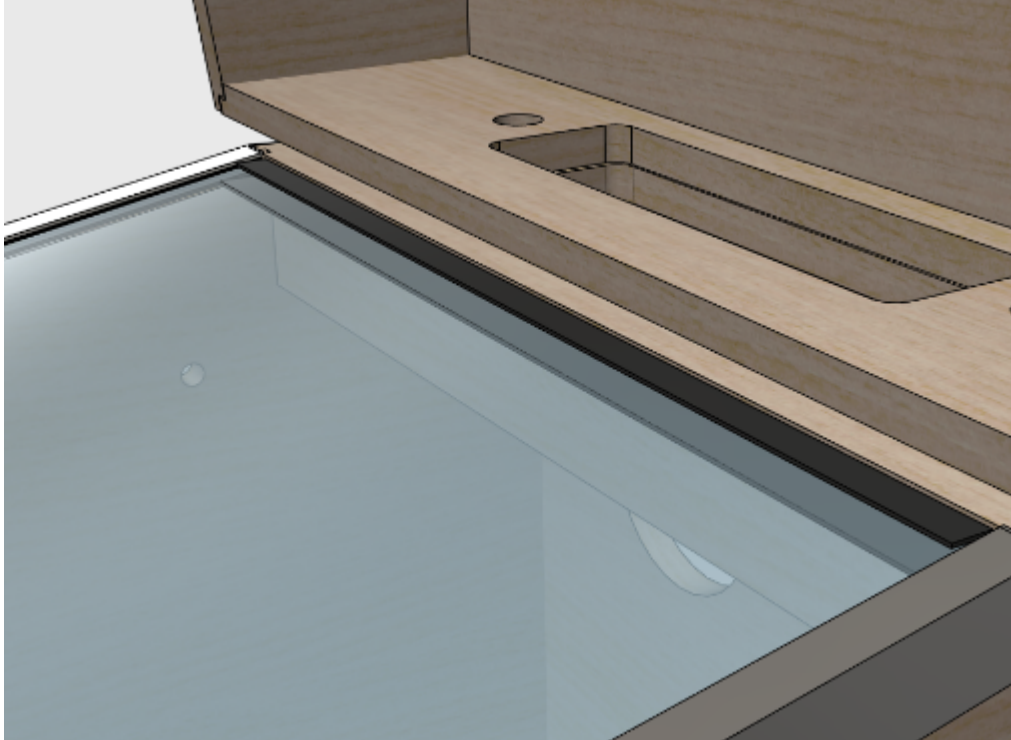
Assuming you're going through this section in order, you've already installed the glass channels that go under the side rails. If not, you should go back and do that before proceeding.

After the side glass channels are installed, there's one more part to install for the top glass, which is a piece of plastic trim at the back of the machine. This provides a slot for the rear edge of the glass to fit into.



This piece of trim attaches to the "shelf" at the back of the cabinet with a few screws, so it's pretty simple at that level (no wacky new router bits required!). But I found it a bit tricky to get the alignment right when I installed it on my machine.

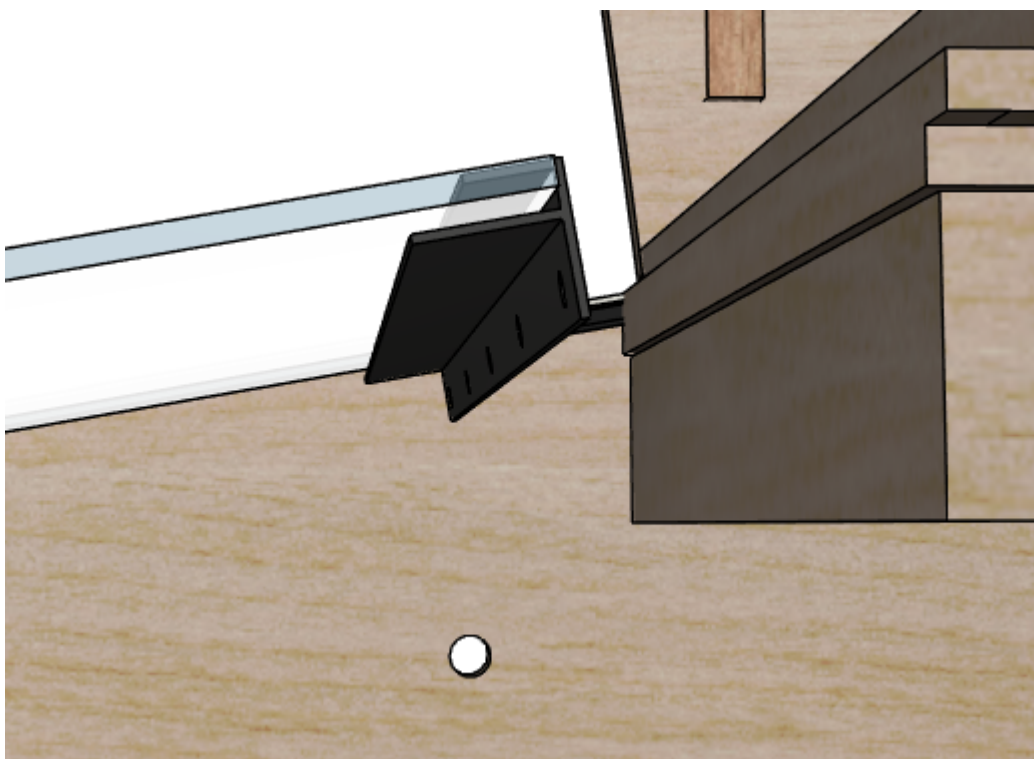




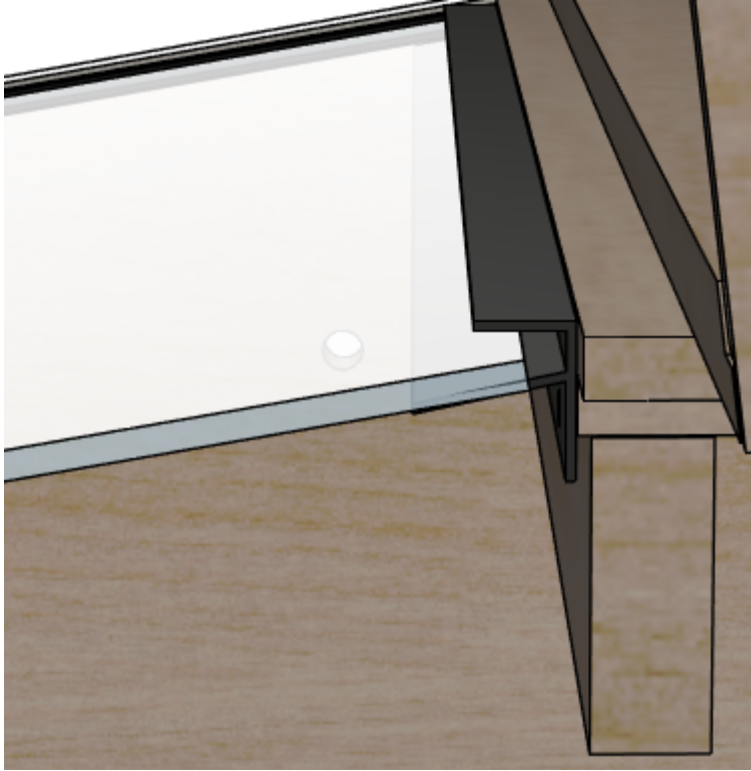
The thing that makes it tricky to install is that the opening in the trim for the glass is just about the same thickness as the glass, so there's not much room for error in aligning it. If the trim isn't aligned perfectly with the glass, the glass will snag on the edges of the trim when you try to slide the glass into place.

The procedure I used was to try to position the trim using the glass itself as a guide. Like I said, I found this a bit difficult in the execution, but I don't have any better ideas.

- Slide the glass into the side channels, almost all the way to the back
- Fit the trim onto the back of the glass, orienting it as shown below

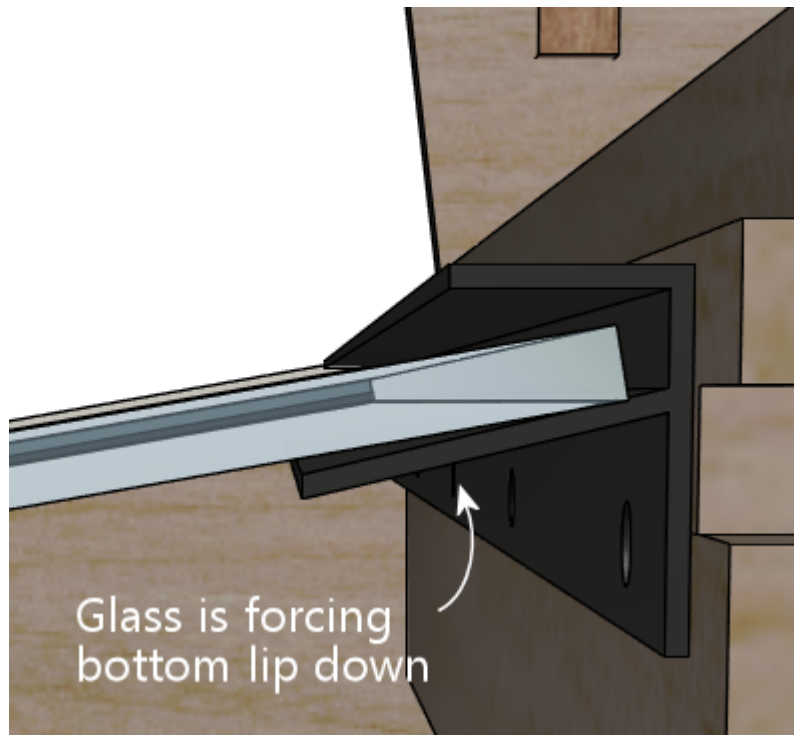


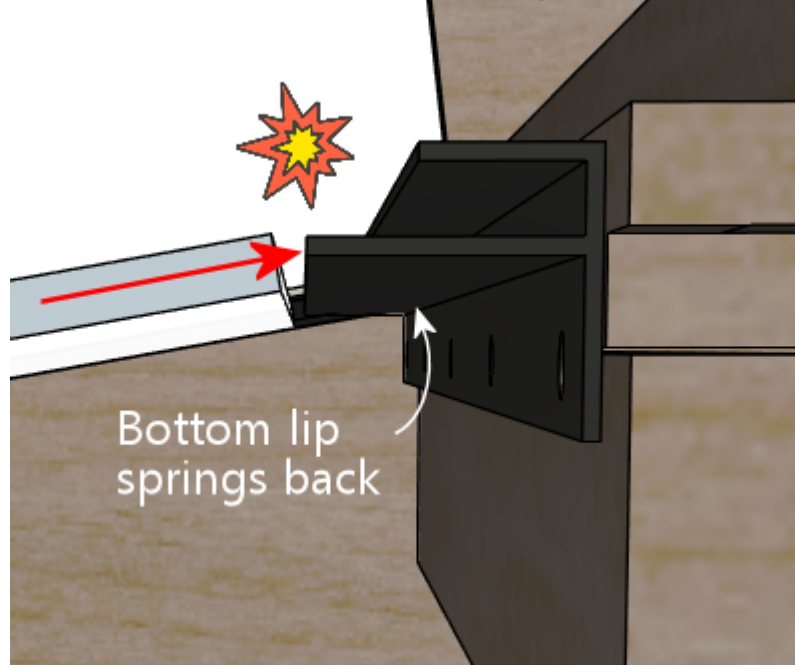
- Slide the glass all the way back, pushing the trim flush against the back shelf



- Mark the position of the trim on the rear wall
- Remove the glass

At this point, the obvious thing to do is to put the trim back at the marked position and screw it into the shelf with some wood screws. That's indeed how I proceeded. The problem I had is that the position we marked above had the glass already in place, and the glass tends to apply a little pressure on the bottom lip that tilts it down slightly. If you install it at exactly this position, the bottom lip of the trim will spring back up without the glass there, so when you try to insert the glass, it'll get hung up there.





I ended up just iterating this a few times with test installs before I found the magic spot. Before you commit to a position, try testing the proposed location with something to hold it in place temporarily, such as masking tape, or a trustworthy assistant. Slide the glass in and out at the test position. Adjust until you find the spot where the glass will slide in smoothly.

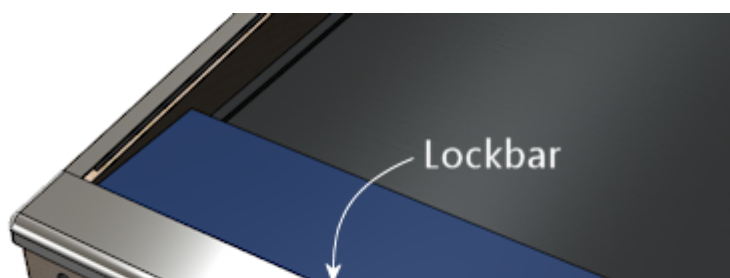
Once you find the right spot, fasten the trim to the rear wall with wood screws. #6 x $\frac{3}{4}$ " should work.

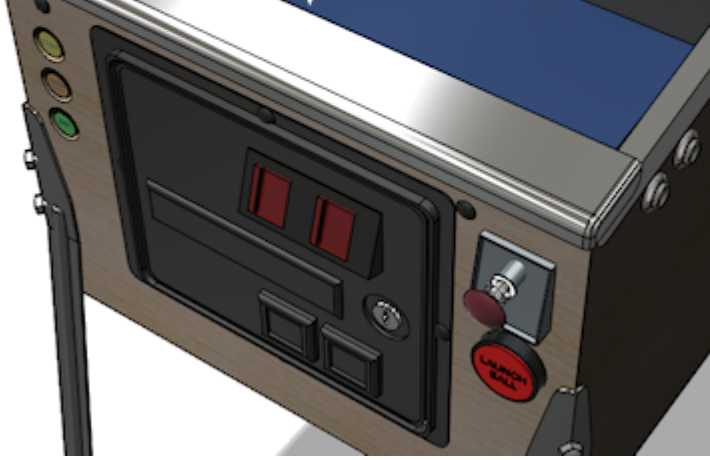
For what it's worth, the install positions on the real machines I've looked at vary from the top of the trim being flush with the top of the shelf, to being as much as $\frac{1}{4}$ " above the shelf. So maybe there's not a mathematically predictable position, as it seems that even the pros resorted to ad hoc alignment. I think this was especially hard on my virtual cab build because my shelf was cut square at the front edge - I didn't use the 10° bevel angle that's recommended in the Chapter 21, cabinet body plans. With the square front edge, my plastic trim piece wasn't aligned properly with the slope of the glass cover. You'll probably have an easier time with this than I did if you used the bevel angle.

Lockbar and receiver

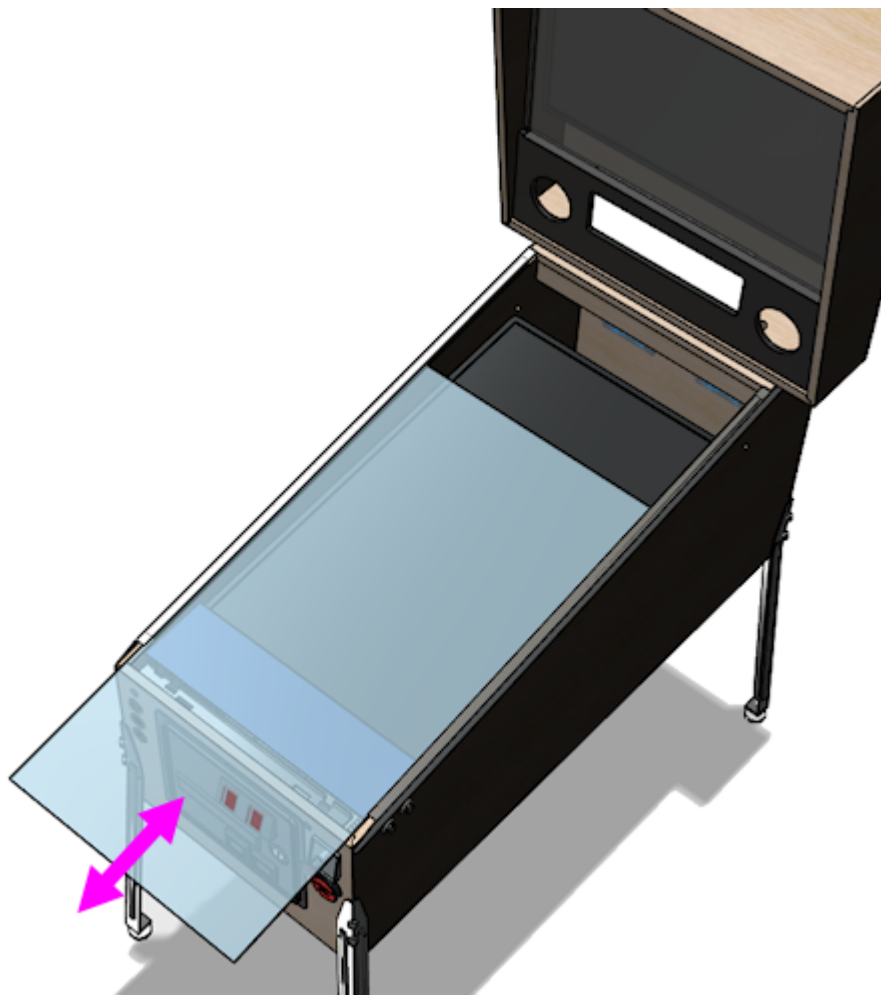
The "lockbar" is the metal trim piece at the front top of the cabinet, so named because it serves to lock the glass cover in place. You'll also see it called a "lockdown bar" and a "lock bar" and various other variations on the "locking" theme. The vendors use all of these terms, inconsistently, so you might want to try them all if you're searching to buy one online.

The "official" name that appears in the Williams parts books is "front molding assembly", so that's another search term to try when shopping.





The lockbar (the name we'll settle on here) serves three main purposes. The first is the locking function that's right in the name. The bar serves to lock the top glass in place, by preventing the glass from sliding out the front. If you want to remove the glass, you first have to remove the lockbar. The lockbar itself is secured by some latches inside the machine, which can be engaged and released via a lever you can reach by opening the coin door. So you can't take off the glass without removing the lockbar, which you can't do without opening the coin door, which you can't do without the keys.



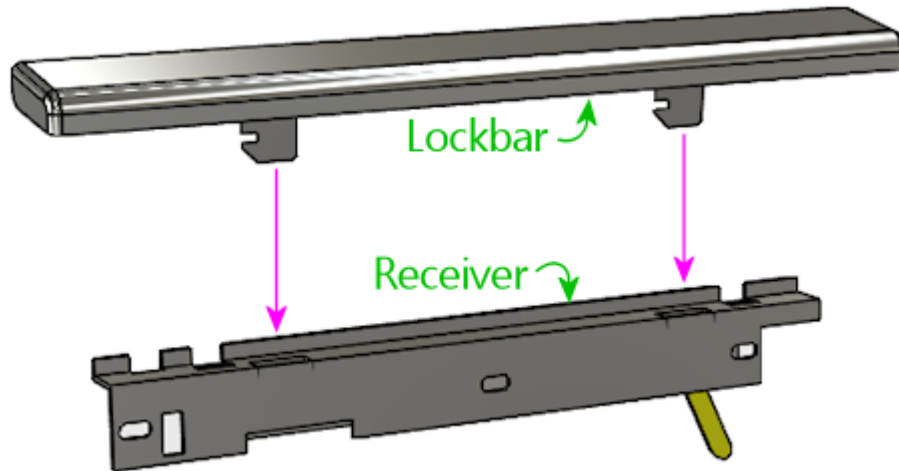
The second function of the lockbar is cosmetic. It serves as decorative trim along the top front edge, as suggested by the official Williams name for the part, "front molding assembly".

The third function is to provide a comfortable place to rest your hands while playing.

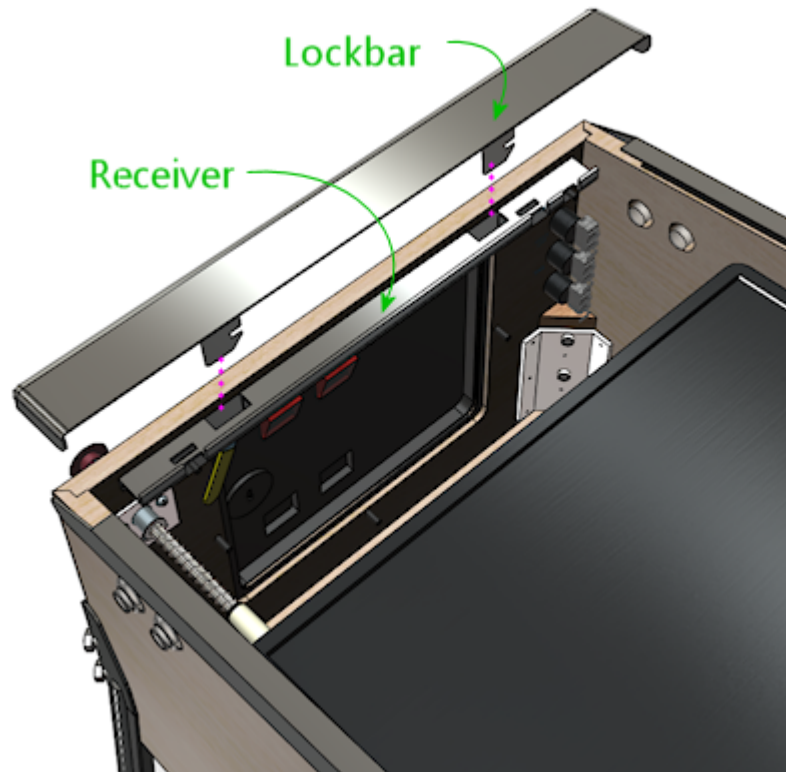
The natural hand position while playing is to grip the front corners of the machine with your fingers on the flipper buttons. The lockbar has nice rounded corners right where your palms go. This hand-rest function becomes apparent the first time you try playing a round of pinball on a machine without the lockbar installed - the bare plywood corners can be awfully sharp.

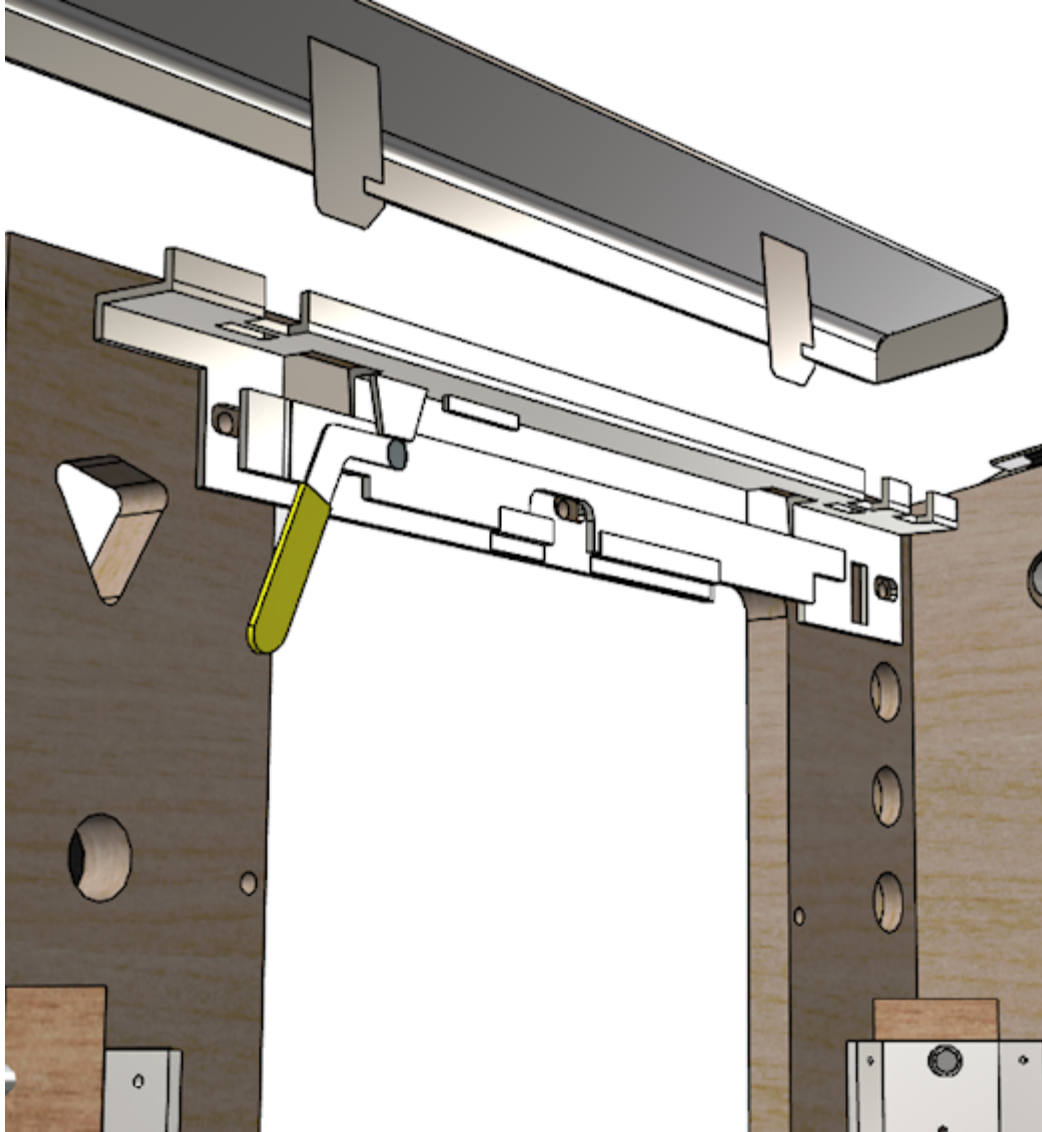
If you're not planning to use a genuine pinball lockbar, you should come up with a substitute that at least provides a comfortable hand-rest, and, if you're using a glass cover, that prevents the glass from sliding out.

In the standard setup, the lockbar mates with another part, usually called the *lockbar receiver*. (The official part name that appears in the Williams manuals is "lever guide assembly".) The receiver attaches to the inside of the machine, at the top of the front wall, and isn't visible to players.



The receiver is installed at the top of the front wall, on the interior side:





Fire button

Many of the 2000s Stern machines feature a button on the top of the lockbar, usually labeled "Fire" or something similar. The button typically activates special features in the game, so it's another interactive game control on par with the flipper buttons and plunger.



You don't have to install a physical Fire button on your virtual cab to play the Stern games that feature a Fire button, since the Visual Pinball re-creations always provide a substitute control that you can use instead, usually the MagnaSave buttons (see

Appendix 4, Tables with MagnaSave Buttons.) But some people like to install a dedicated lockbar Fire button anyway, since it replicates the playing experiences more faithfully for tables that featured the button in the original arcade version. If you're a big fan of the more recent Stern titles, it might be worth including a physical Fire button on your lockbar.

The simplest and surest way to install a Fire button is to buy a Fire-button-ready lockbar and receiver made for the Stern machines. Those parts are specifically designed to accept the button, so installation is straightforward. You can also adapt regular lockbar parts to use a Fire button, but it's more work - we'll explain how below.

Option 1: Use Stern parts designed to include the Fire button. This is the simplest approach, since you don't have to modify any of the parts.

- To find a lockbar that can accommodate the button, the search term that seems to work best is "premium lockbar", because Stern typically only includes the extra button as an added feature on the upgraded versions of their games ("premium" or "limited edition"). One example: lockbar for *Star Trek Premium*, Stern part 500-7283-22.
- The receiver that's compatible with a center button is Stern part 500-7237-00
- The button itself is an extra-long (1-3/8") clear flipper button, Stern part 515-7791-00
- Button collar (mounted on top of lockbar), Stern 545-7292-10
- Mounting plate (mounted under lockbar), Stern 545-7291-00
- Palnut (secures button to lockbar), Stern 240-5003-01, Williams/Bally 02-3000
- #8-32 Keps nuts, quantity 2 (secures mounting plate), Stern 240-5104-00

As far as I can tell, there's no such thing as a "generic" lockbar-with-button. They're all made for specific games (*Star Trek*, *Lord of the Rings*, *Game of Thrones*, *AC/DC*, etc), and all of the ones I can find come with powder-coat finishes (not the standard chrome) and special game-specific badges. The game-specific badge in particular would be a big negative for me, in that it would clash with my custom theming, but it's actually a separate part that you could remove and replace with something custom. You'd also almost be forced to use the matching powder-coat finish on the legs and side rails. That might be something you want anyway, as it can look snazzy, but it would increase the cost for those parts. And finally, keep in mind that these lockbars are only available in the standard-body cab width, so these wouldn't be an option if you're building a widebody or custom size.

Feedback request: I'd sure like to know if there are any **generic** lockbar-with-button options (*with* the button hole, in the standard *chrome finish*, and *without* any game-specific badging). Please pass along a pointer if you know of such a product available commercially. Also, if you've personally modified a *regular* lockbar and receiver combo to include a Fire button, I'd like to hear about how you did that and how well it turned out. I'd be thrilled to have detailed conversion plans to add to this section. The options above seem regrettably limiting.

Option 2: Add a Fire button to a regular lockbar. It's possible to install a Fire button in a standard Williams lockbar and receiver, but you have to modify some of the parts and do some custom assembly work.

Parts:

- Transparent flipper button, 1-1/8" length, part A-16883-13

- Pushbutton mounting spacer, part 545-7292-10
- Pal nut, Part 240-5003-01
- Use my custom board:
 - Grab these plans and fabricate them at OSH Park: mjrnet.org/pinscape/downloads/Lockbar-Fire-Button-LED-plus-switch.zip
 - Switch: DTS-62K-V
 - LEDs: Kingwin WP154A4SEJ3VBDZGC/CA, quantity 2 (or any other 5mm common-anode RGB LED)
 - 100 Ohm resistors, 2mm x 7mm size, qty 2 ("R" resistors)
 - 47 Ohm resistors, 2mm x 7mm size, qty 4 ("B" and "G" resistor)
 - Molex 22-05-3071 connector
 - Generic 0.1" crimp-pin wire housing, 7 pin positions
- **Or** use a generic membrane switch (search Amazon or eBay for **single key membrane switch**) and any LED that you can fit into the space

Step 1: Drill a hole in the center of your lockbar the same diameter as the **stem** of the flipper button (typically 5/8", but measure yours to be sure). I haven't tried this myself, but the advice I've heard is to use a stepped drill bit. Drilling metal usually works better with the drill at low speeds, and a drill press is better than a hand drill.

Step 2: Place the spacer on the button, insert the button in the lockbar hole you just drilled, and secure on the back with the Pal nut.

Step 3: Connect the LED and switch.

If you're using a membrane switch and a separate LED, you'll have to improvise mountings for them. You should attach everything to the receiver, not the lockbar, so that you can easily remove the lockbar. Make sure that all of the wiring connections are insulated so that nothing shorts if it comes into contact with the metal parts in the receiver or lockbar.

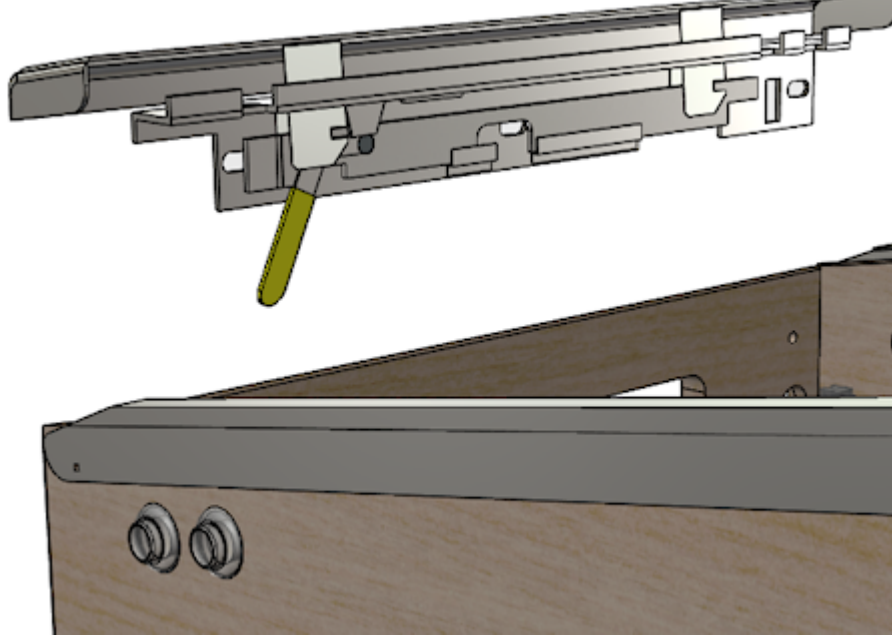
If you're using my circuit board, solder the parts, then attach it to the receiver (not the lockbar) directly behind the button. It should just fit into the space between the lockbar and receiver, so I think you can get away with a simple approach like taping it to the receiver with something strong, perhaps electrical tape or duct tape. Make sure there's a layer of insulation between the board and the metal receiver parts, to prevent shorts. Electrical tape will work for this, but something like "fish paper" would be better.

Build the wire housing and wire it to the control boards. The terminals marked **SW** connect to the switch, so connect these to your key encoder (connect one to the key encoder "Common" or "Ground" terminal, and connect the other to the switch input you've assigned as the Fire button). Connect the pin marked **+** to a +5V power supply. Connect the pins marked **R**, **G**, and **B** to your DOF output device controller ports for the Red, Green, and Blue segments of the Fire button light.

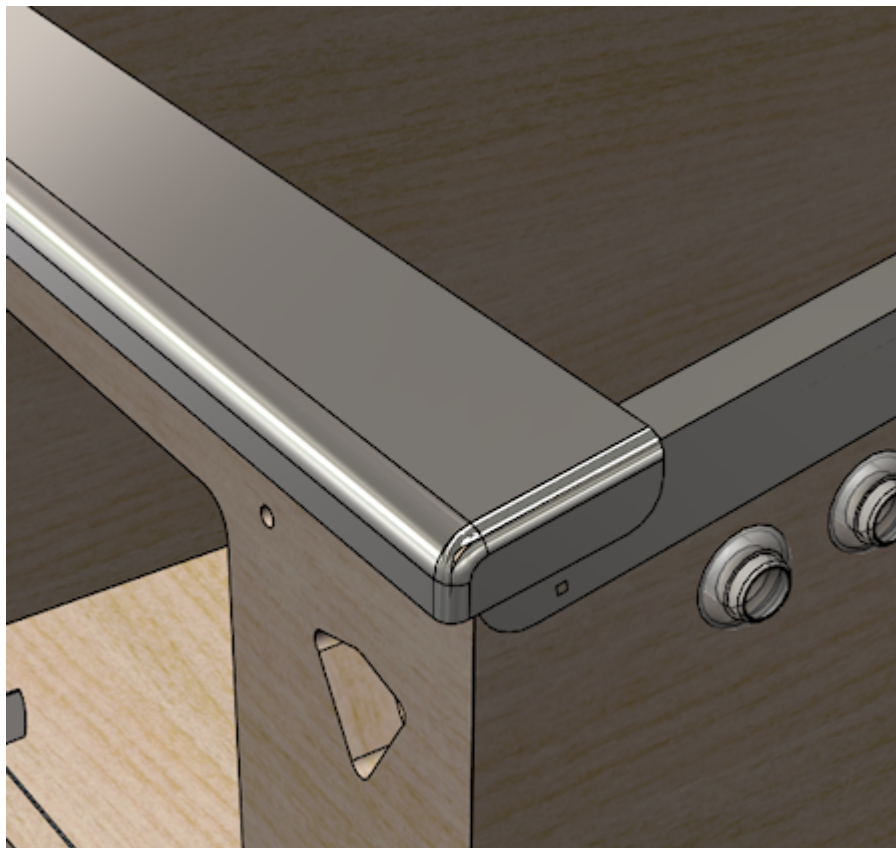
Installing the lockbar receiver

Before you install the lockbar receiver, install the side rails, as described earlier in this section. The lockbar should fit snugly on top of the side rails, so the rails have to be in position before you can fine-tune the lockbar positioning.

Start by fitting the lockbar into the receiver (with nothing installed in the cab yet).

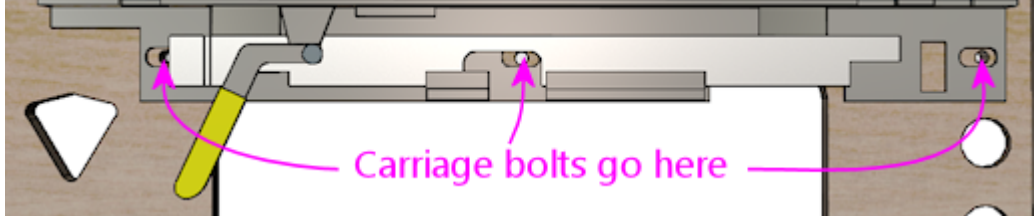


Put the lockbar/receiver assembly into position. On the inside of the cab, the vertical part of the receiver should be flush with the front wall. On the outside, the lockbar should be resting on the side rails, slightly overlapping their front edges so that there's no gap. The front of the lockbar should overhang the front wall of the cab slightly (by about 1/8" to 3/16").



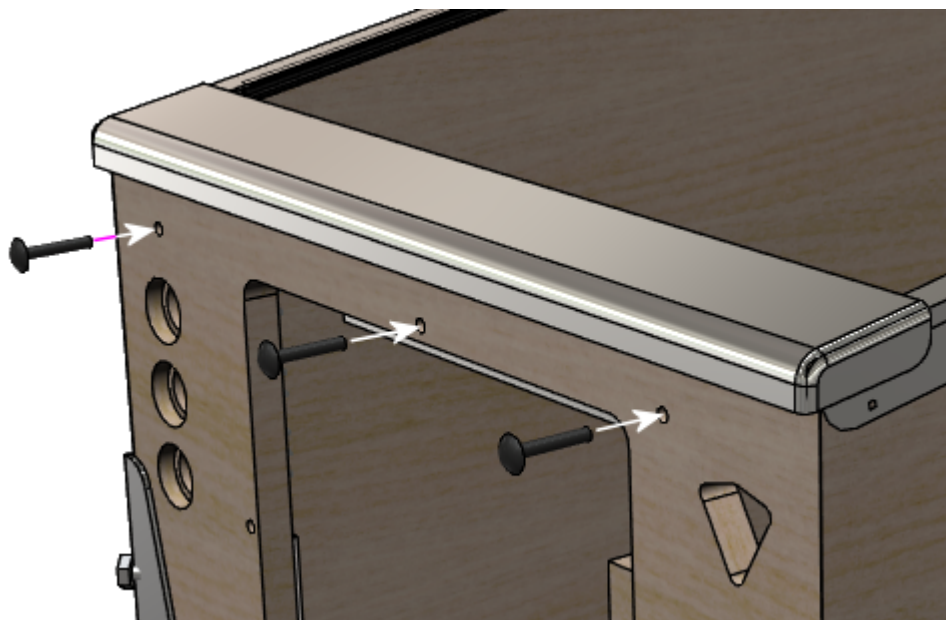
If you haven't already drilled the holes in the front wall for the receiver's three carriage bolts, mark the center positions of the bolt holes in the receiver on the inside wall. The positions of the bolt holes are illustrated below. After marking the locations of the holes, remove the lockbar/receiver assembly and drill them at 5/16" diameter. Put the assembly back in place.





The receiver attaches with three ¼-20 x 1¼" carriage bolts and ¼-20 lock nuts. These are available in silver or black finishes. Most of the real machines use black bolts to make them less conspicuous. I've only been able to find them in black from the pinball supply vendors (Pinball Life, Marco Specialties) - they're not just painted black, but actually have a black oxide finish.

Insert the carriage bolts from the outside of the cabinet:

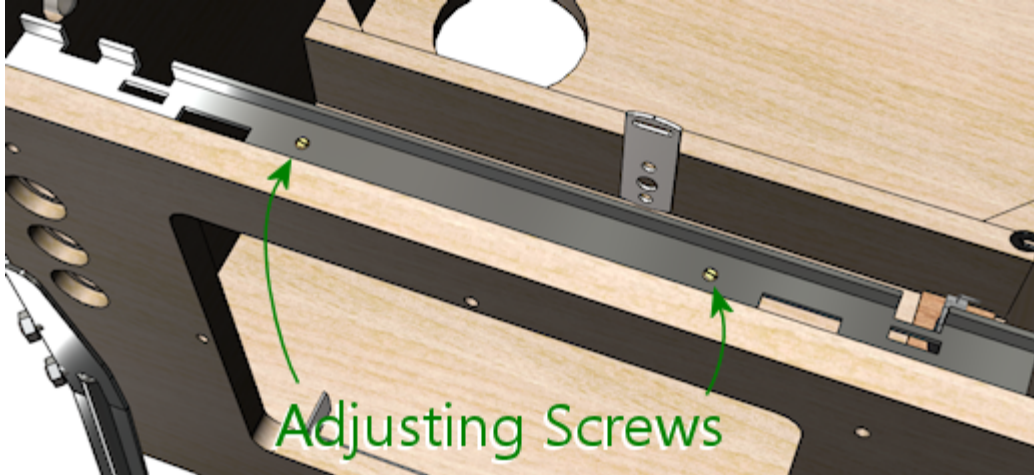


Attach lock nuts on the inside. You might need to pull the lever on the receiver to access one or more of them, since parts of the mechanism can slide in front of the bolt holes. If it's too hard to fit the nuts onto the bolts with the lockbar in place, remove the lockbar for that step. I'd put it back in place before final tightening, though, to make sure things stay properly aligned. The receiver provides a little bit of play in the bolt holes to let you fine-tune the position, and the best way to do that is to use the lockbar itself as the guide.

Note that the center bolt is shared with the coin door, so you should leave that out for now if you're going to install a coin door next. I'd still insert the center bolt during the fitting process to make sure the holes are all properly aligned, but don't actually fasten it yet.

Check final alignment by removing the lockbar and then putting it back in place. You should be able to smoothly remove it and re-attach it, and it should still be sitting at the desired position when latched in place again. If it's too tight or too loose, you can loosen the bolts again and tweak the receiver positioning to improve the fit. The receiver has oversized bolt holes to give you a little play to get the position right.

You can also adjust the locking tension slightly via the two brass adjustment screws on the top of the receiver, as illustrated below. Tighten the screws (turn them clockwise) to push down on the latches and increase the tension when the lockbar is installed.



DIY alternatives to real pinball lockbars

Some pin cab builders don't use real lockbars because of the cost, or because they're building an unusual cab design where the standard lockbar doesn't fit the style or the available space.

If size (not price) is the only factor, note that you can buy lockbars in custom widths (made to your specifications) from VirtuaPin.

Fashioning your own metal lockbar seems like a challenging job for a DIYer, short of having access to a well-equipped metal shop. I'm afraid I don't have any workable suggestions here; it's not the sort of thing you can make on a 3D printer, which is the magic answer to almost everything else these days. The closest starting point might be an "L" bar, which you can buy from hardware stores in various metals and thicknesses - but I'm not sure how you'd mold that to the more complex shape of a standard lockbar with its rounded corners at each side.

If you're going for a furniture look with wood trim all around, it's possible to craft a wood version using fairly ordinary wood-working tools. Here are some vpfforums threads that might be helpful:

- [Wooden custom lockdown bar](#)
- [Alternative for a custom lockdown bar](#)

Another possibility is to use a 3D printer to fabricate a plastic lockbar. Here's a vpfforum thread about that, with advice about materials and finishes to make it look like a metal lockbar:

- [Alternative for a custom lockdown bar](#)

DIY alternatives to a real lockbar receiver

To save a little money, some virtual cab builders omit the receiver, even while using a real lockbar. The receiver is a purely internal part, so it doesn't serve any cosmetic function, and some pin cab builders find the price (currently about \$80) unreasonable for a part with such a simple job. The main thing that makes a standard receiver so expensive is that it has to be rather heavy-duty to fulfill its role as a security lock. For a home machine, you might not be concerned about teenagers trying to pry the thing apart to steal quarters.

One simple solution might be to use Velcro. You'd have to attach some filler material to the bottom of the lockbar to fill the space between the lockbar and the top of the front wall. Once the two areas are in contact, you can simply glue a bunch of Velcro to each side. This would hold the lockbar reasonably well, although obviously not in a truly "locking" way, and it would probably feel a bit wobbly compared with the real

ones.

A more elaborate home-made alternative is described here:

Question about lockdown bars and receivers for mini cabinet (message #5)

Briefly, the idea is to place a pair of toggle latches on the inside front wall of the cabinet, positioned to align with the hooked prongs that stick out of the bottom of the lockbar. To fasten, you reach in through the coin door, hook the latches to the prongs, and engage the latches. To release, you again reach in through the coin door and disengage the latches. The downside is that it would require a fair bit of dexterity to reach the latches through the coin door, since they need to be positioned around the corner on each side. In contrast, the standard receiver can be engaged and disengaged with a lever that's positioned within easy reach.

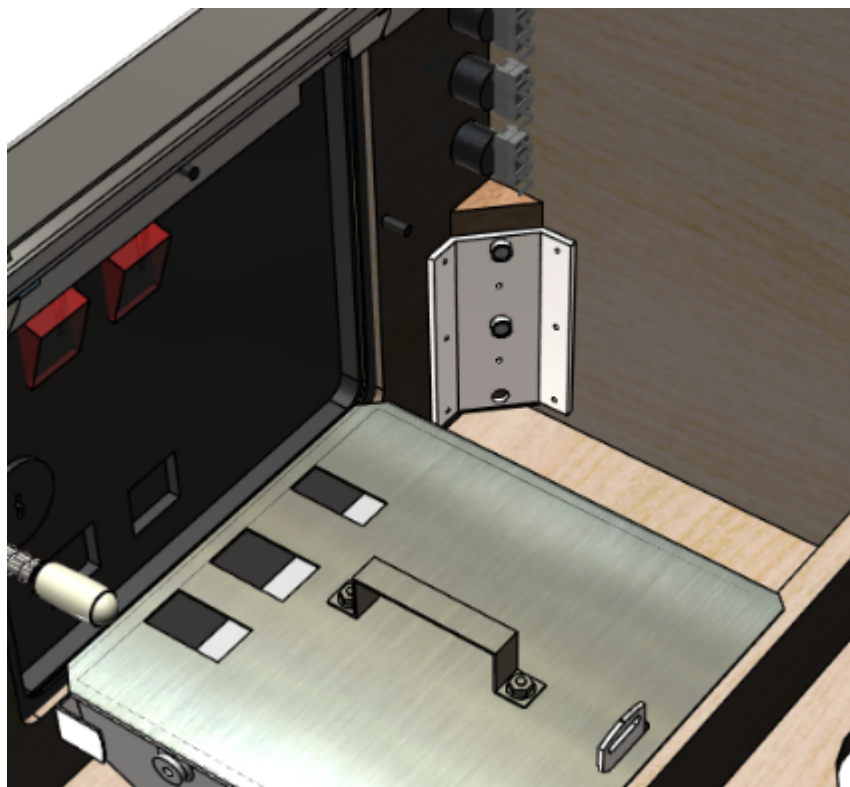
Note that some of the newer Stern machines actually use lockbars with a similar toggle-latch design. Compatible Stern lockbars are equipped with under-carriage latch-hook parts that are specifically designed to be used this way, so you might find it easier to use this approach with a compatible Stern lockbar than with a lockbar designed to fit the standard Williams/Bally receiver. Refer to these parts:

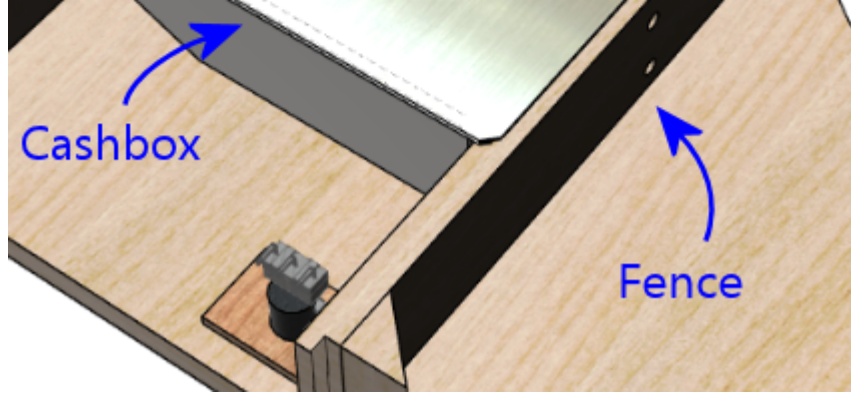
- Stern lockbar, dual luggage latch, 5500-6882-03-00
- Luggage latch, 355-5038-00

Cashbox

If you're planning to drop coins into the coin slots, you'll need something to catch the coins on the other side. You don't want them rolling around loose where they could randomly short out wiring or get wedged in something mechanical.

The real machines' solution is the "cashbox". It's a low-profile plastic box with a metal lid, with slots in the lid that line up with the coin chutes. It sits just inside the coin door. When a coin goes through one of the chutes, it drops straight into the cash box. The cash box is sized to fit through the coin door, so the operator can easily collect the machine's income when making rounds.



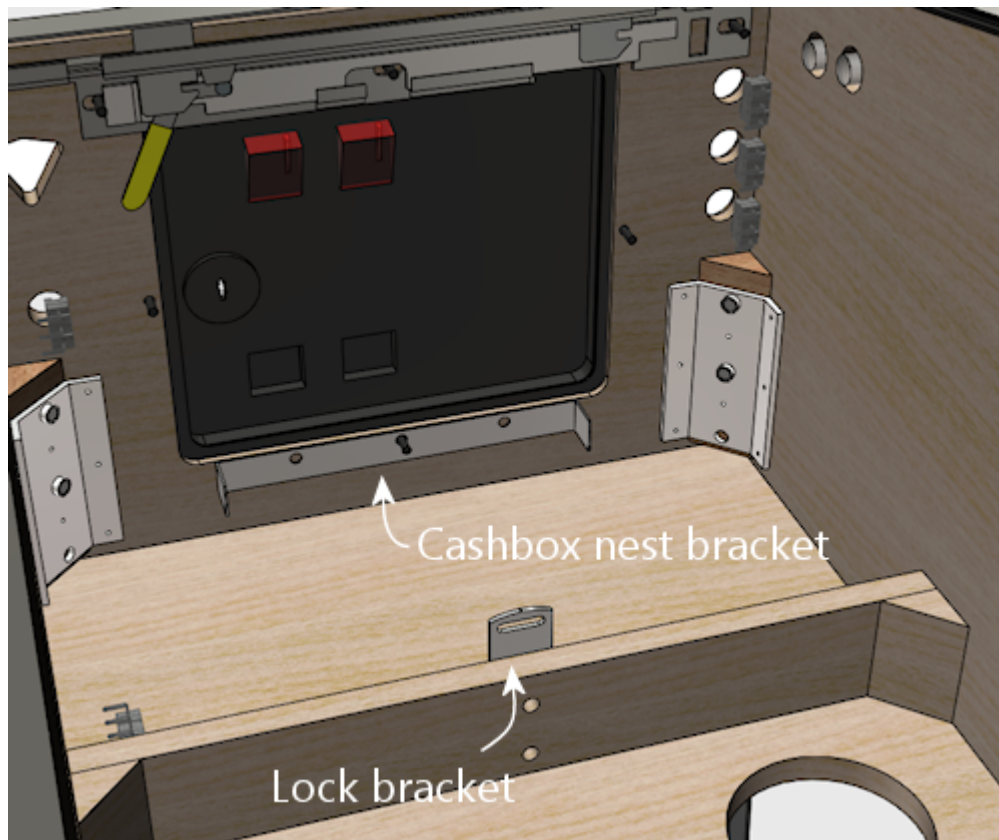


The cashbox has essentially just one design these days, which looks like the illustration above. Older machines used a variety of shapes and sizes, but nearly all pinballs made since the 1990s use the same design that Williams used in the WPC machines. Since it's so close to a standard, it's the one you can buy from pinball vendors. Most of them sell it in two separate pieces: a plastic tray, and a metal lid. They don't separate them just to make your life difficult; it's for modularity, so that the same tray works with lids with different slot patterns for different coin door layouts. The three-slot lid illustrated above is for the typical US coin door configuration. If you have a non-US coin door, you should be able to find a matching cashbox lid at a European pinball parts vendor.

Installation

The cashbox itself doesn't require installation per se; you just pop it into the space at the front of the machine. But you do have to install two brackets to hold it in place, plus a little "fence" or divider wall, called out in the illustration above.

The first bracket goes at the front of the cab, directly under the coin door. This is the "cashbox nest bracket", Williams part 01-6389-01. It prevents the box from sliding back and forth.



The nest bracket has three screw holes. The center one is meant to align with the bottom bolt in the coin door, so that you share the same bolt between the coin door and this bracket. If you haven't already installed the coin door, slip a carriage bolt ($\frac{1}{4}$ -20 x $1\frac{1}{4}$ ") through the hole from the front of the cab for alignment. (If you've already installed the coin door, just remove the nut from the bottom bolt.) Slip the nest bracket's center hole over the bolt to position the bracket. Make sure it's level, then fasten the two outside holes to the cab's front wall with wood screws (#6 x $\frac{3}{4}$ " should work).

If you've already installed the coin door, reattach the nut on the center bolt. Otherwise just leave that off (and take the bolt back out) for now; you'll install it when you get to the coin door.

The second bracket is the "cashbox lock bracket" (Williams part 01-10030 or 1A-3493-1), which attaches to the fence called out in the earlier illustration. If you followed our plans from Chapter 21, *Cabinet Body*, you've already installed that. If not, you should go back now and follow the plans in that chapter under "Cashbox fence".

Once both brackets are in place, installing the cashbox is a simple matter of dropping it into the space delineated by the fence, fitting the slot at the back of the cashbox lid over the lock bracket. To remove the cashbox, lift the back edge high enough to clear the lock bracket, and pull the cashbox out. This is all meant to be done through the coin door, since the cashbox is sized to fit through the door.

(If you look more closely at the lock bracket, you'll see that it has a little slot at the top. That's for attaching a padlock, to add an extra layer of security to protect the booty even if someone gets past the coin door. Probably not something you'll need in a home machine.)

DIY cashbox

Apart from cost, the main reason you might want to consider designing your own home-made cashbox substitute is that the real ones are rather large. The standard cashbox is great at its job, but it takes up a whole foot of floor space at the front of the cab, which impinges on space you might want to use for PC parts or feedback devices.

Improvising a home-made cashbox isn't too challenging, since it's just a box with a couple of holes in the lid. You could easily fashion one out of plywood or acrylic. I created one using a plastic food container; I found one with about the right depth, and used an X-acto knife to cut slots in the lid that line up with the coin chutes. I use a bungee cord (connected to a couple of eyelets screwed into the floor) to hold it in place. It's certainly not as elegant as the real cashbox (particularly the bungees pinning it down), but it only takes up about 5" of floor space.

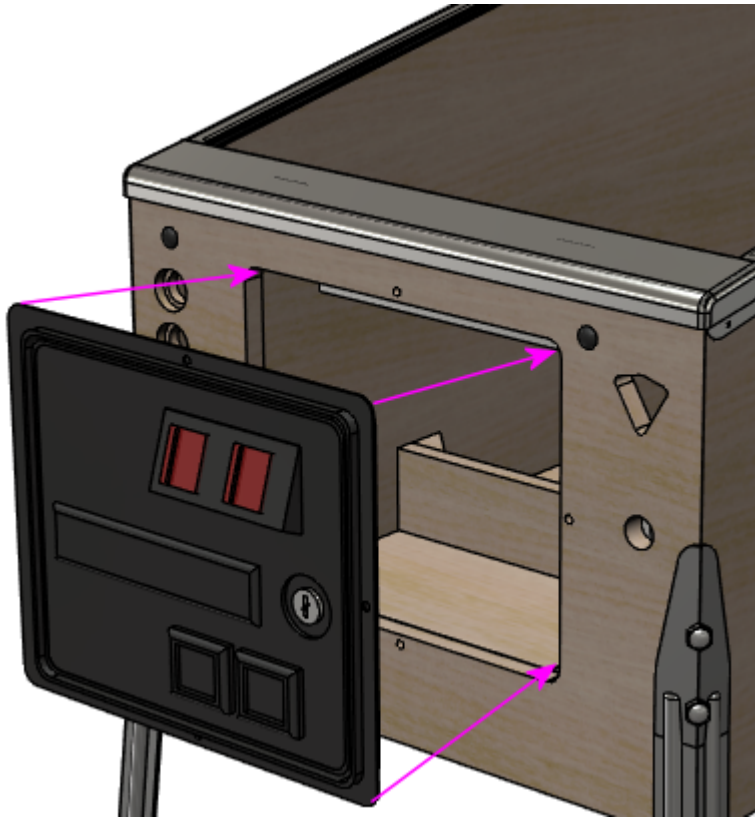
Coin door

The coin door is a complex enough subsystem that we've devoted a whole chapter to it (Chapter 40, *Coin Door*). But we'll go over the basic installation process here.

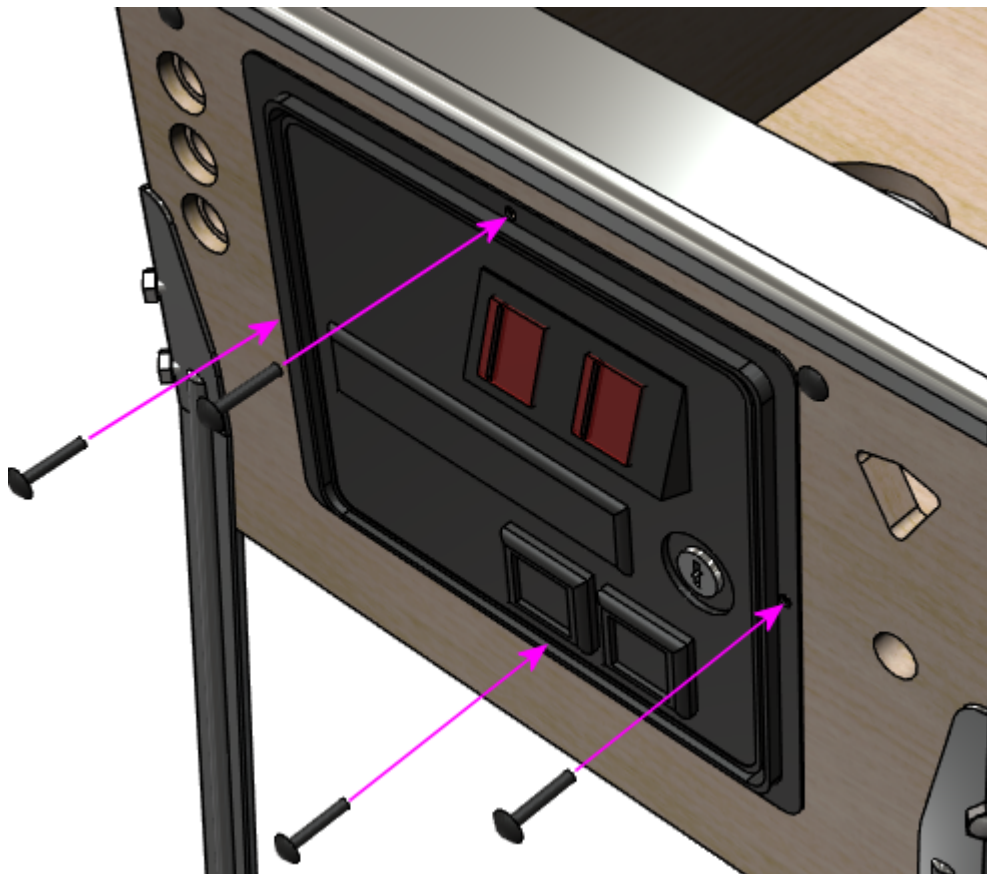
If you're using a standard lockbar-and-receiver combination, it's easier to install the receiver before installing the coin door. Follow the procedure above. The receiver shares its center attachment bolt with the coin door, so you'll need to remove the center bolt in the receiver if it's currently in place.

The standard coin door assembly comes with the door itself and the frame already assembled, so there's not much to installing it. Start with the door closed and locked.

Fit it into through the coin door opening in the front wall.



Holding the door in place, insert carriage bolts ($\frac{1}{4}$ -20 x $1\frac{1}{4}$ ") through the four holes around the perimeter of the door frame. Fasten them on the inside with $\frac{1}{4}$ -20 lock nuts.



The top bolt in the coin door is shared with the lockbar receiver, if you're using one.

Thread the bolt through the matching hole in the receiver mechanism, and attach the lock nut on the interior side of the receiver, so that it secures the receiver as well as the coin door.

If you're installing the full set of cashbox hardware, the bottom bolt in the coin door frame will be shared with the cashbox "nest bracket". Thread the bolt through the matching hole in the nest bracket and attach the lock nut on the interior side of the bracket.

Coin mechs

If you bought a brand new coin door, it probably didn't come with coin "mechs" (mechanisms), the gadgets that sit behind the coin slots to validate inserted coins. You can buy those separately. The mechanical quarter acceptor used in typical US coin doors is an inexpensive add-on (about \$10 each). I think it's worth including these in a virtual cab, for the added realism of being able to use real coins. The installation procedure is detailed under "Coin mechs" in Chapter 40, Coin Door.

Custom coin slot inserts

On most types of coin doors, it's possible to replace the illuminated "25¢" signs (known as "inserts") on the coin slots, to show different coin denominations, or better yet, your own custom graphics. See "Custom coin slot inserts" in Chapter 40, Coin Door.

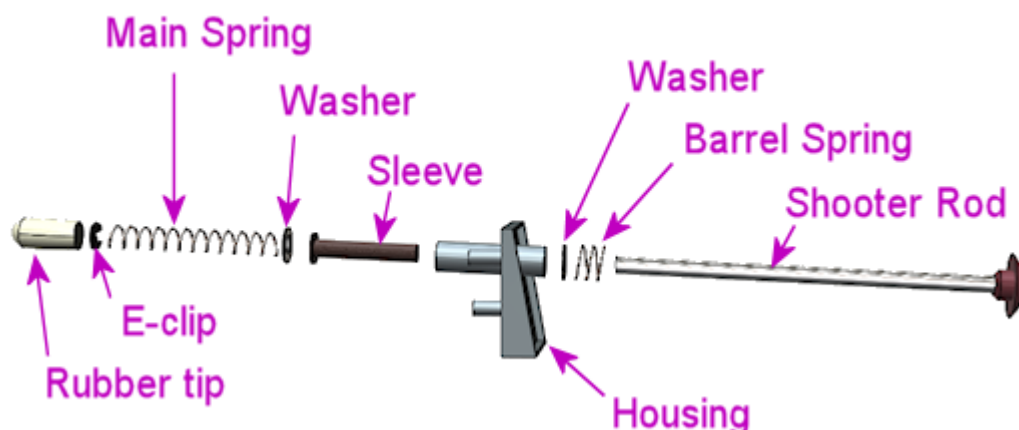
Coin door position switch

On real pinball machines, there's a switch inside the cabinet that detects whether the door is open or closed, just like the light switch in a refrigerator door. It's useful to include this switch in a virtual cab, because many ROM-based tables won't let you access the setup menus unless they get a signal from the switch indicating that the door is open.

Full instructions on setting up the door switch (as well as connecting it to the virtual pinball software) can be found in Chapter 40, Coin Door, under the section "Coin door position switch".

Plunger

If you bought a full plunger assembly, it probably came assembled. If not, or if you bought the separate components, assemble as shown below.

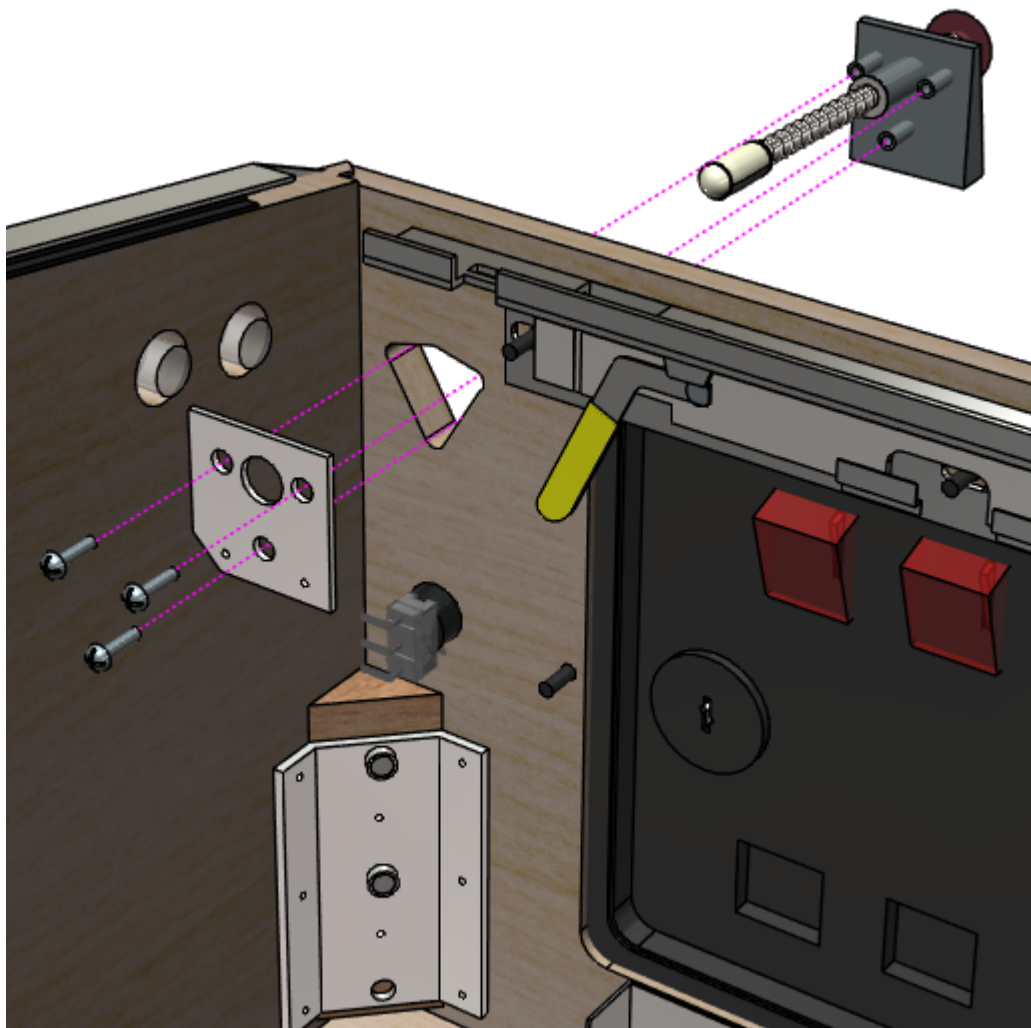


- Slip the barrel spring over the shooter rod and push to the knob end

- Slip the washer over the shooter rod and push down to the barrel spring
- Insert the nylon sleeve into the shooter rod opening in the housing (from the inside of the housing)
- Insert the shooter rod into the opening the housing (from the outside of the housing)
- Slip the other washer onto the shooter rod
- Slip the main spring onto the shooter rod
- Attach the E-clip to the rod. You'll have to hold the spring back while you do this, since the spring will be compressed in its normal position. The E-clip fits into the groove near the end of the rod. Use needle-nosed pliers to snap it into position.
- Fit the rubber tip over the end of the rod. (This is optional in a virtual cab; you probably don't need the tip unless you're using some kind of optical sensor that requires it. Leaving it out will save a little space if you have tight clearance to the TV.)

If you haven't already routed the opening in the front wall for the plunger, see "Plunger and Launch button" in Chapter 21, Cabinet Body.

For installation in the cabinet, you'll need three #10-32 x $\frac{5}{8}$ " machine screws ($\frac{3}{4}$ " length will also work) and a ball shooter mounting plate (Williams/Bally part 01-3535). You can improvise something to replace the mounting plate if you prefer, but the plate makes things a lot easier and only costs about \$2.



Fully assemble all of the plunger parts (except for the mounting plate) as described

above, then fit the assembly through the triangular opening in the front wall, from the outside. The three prongs in the front of the housing should fit in the obvious way at the corners of the triangular cutout. Align the mounting plate on the inside, fitting the large hole at the center over the shooter rod. The mounting plate should sit flush with the front wall of the cabinet. Screw in the three #10-32 bolts.

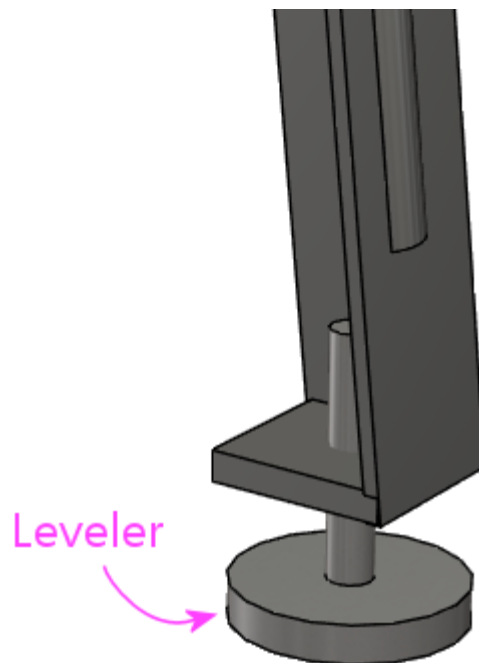
Legs

You'll probably want to leave the legs off for most of the build process. It's easier to install the internal parts (the PC, TVs, buttons, feedback devices, etc) with the machine on the floor or on your workbench. That's why we've saved this for near the end of the hardware chapter. On the other hand, it's easy to attach and detach the legs as needed, so you can always test them for fit.

Assuming the leg brackets are already installed (see above), attaching the legs is pretty easy. You'll need eight bolts (two per leg), $\frac{3}{8}$ -16 by $2\frac{1}{2}$ " or $2\frac{3}{4}$ ". The longer length is usually needed if you have leg protectors of some kind. You should buy the bolts from a pinball vendor rather than using generic hardware store parts, as the pinball bolts look nicer; this is a cosmetic item.

The legs on modern machine are all interchangeable front/back/left/right, so you should have a set of four identical legs. (The front and back legs are the same length. The forward tilt of the machine comes from the back legs being attached lower on the cabinet than the front legs.)

If you haven't already attached the "levelers" (the round foot pads) to the legs, do so before installing the legs. These simply screw in to the holes on the bottoms of the legs.



The levelers let you adjust the slope of the machine slightly, and also let you adjust each leg so that all four legs are planted on the floor (to solve the classic wobble problem with a four-legged table on an uneven surface). It's best to screw all of the levelers all the way in initially (so that they're as "retracted" as possible), then adjust as needed once the machine is situated. The levelers can get a little wobbly themselves at their maximum extension, so keep them retracted and only extend as needed.

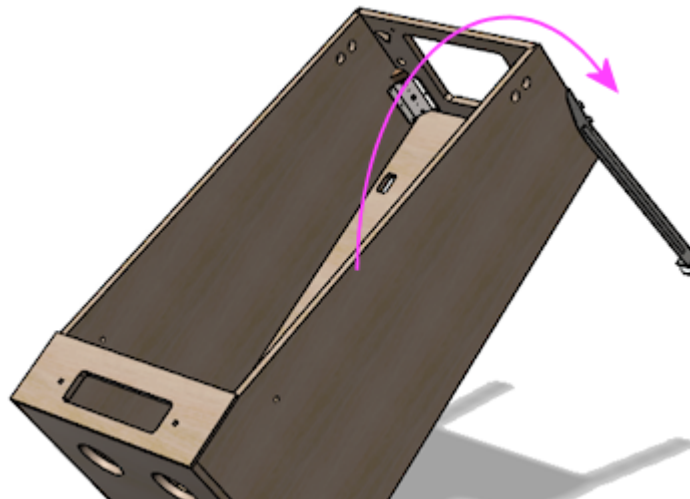
Start by setting the machine on its back. This lets you attach the front legs without any weight on them.




Position each leg at the corner of the cabinet where it goes, aligning the bolt holes in the leg with the bolt holes in the cabinet. If you're using leg protectors, they go between the leg and the cabinet.

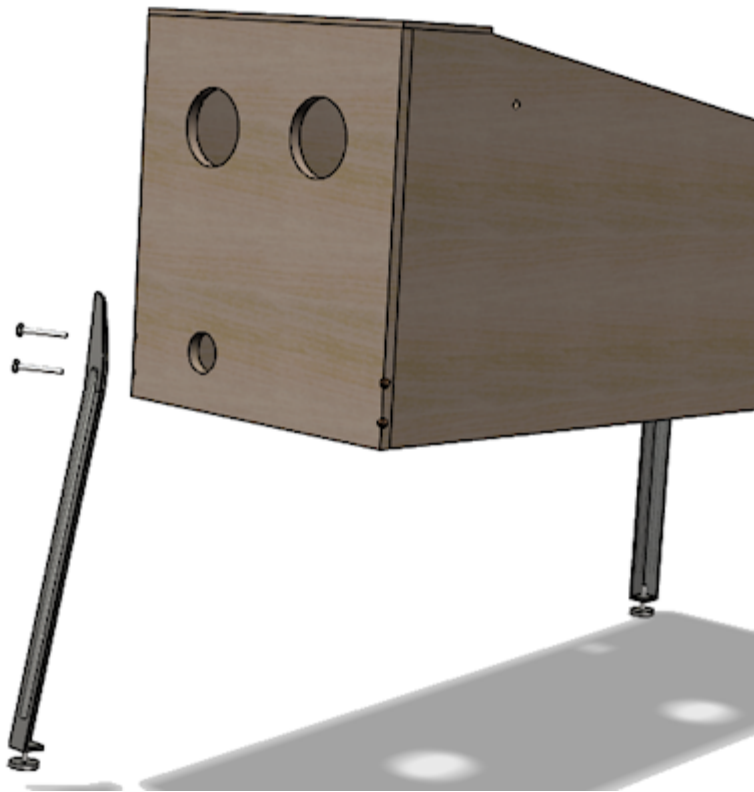
Insert the two bolts and thread them into the bracket. There's no need for any nuts or washers, as the brackets themselves are threaded and serve as the fasteners. Use a hex driver or wrench to tighten the bolts. They should be tight enough that the legs won't wobble, but don't tighten so much that you strip the threads or crush the plywood.

Tip the machine forward until the front legs touch down on the floor, then lift the back of the machine high enough to attach the back legs.





Have an assistant hold the back of the machine up while you install the rear legs, which bolt on just like the front legs. Be sure to have your assistant continue holding the machine off the ground in back until all of the rear bolts are fully tightened.



When the machine is situated at its permanent location, adjust the leg levelers (the foot pads at the bottom) to level the machine, so that all four feet are firmly on the floor without wobble. (On a real machine, you'd also take this opportunity to adjust the leg levelers to fine-tune the cabinet's tilt to level the playfield side-to-side and make its slope match the manufacturer's prescription. But this is superfluous on a virtual cab, where game gravity only exists within the simulation.)

To remove the legs, simply reverse the installation procedure.

Leg protectors: A lot of people use some sort of padding between the legs and the cabinet, to protect the cabinet corners where the legs attach against wear from the pressure and motion of the legs. You can find these on pinball parts vendors by searching for "leg protectors". I've seen two types: felt and metal. The felt protectors will just protect against scratches, while the metal ones help reinforce the whole corner. In commercial pinball machines, it's common to see wear and damage at the leg attachment points, so a lot of collectors consider leg protectors a must. My own experience is that machines in home use don't need them, but they won't do any harm, and they're a relatively inexpensive bit of insurance. One case where I'd consider them more seriously is a cabinet built from MDF, since MDF isn't as strong as plywood, especially for concentrated pressure points like the leg joints.

Protecting decals: With commercial machines, it seems to be a common problem that cabinet decals can wrinkle around the legs, due the pressure that the legs apply against the side of the cabinet and motion at the joint. I haven't seen many reports of this with virtual pin cabs, so it might not be as much of an issue in home-use-only

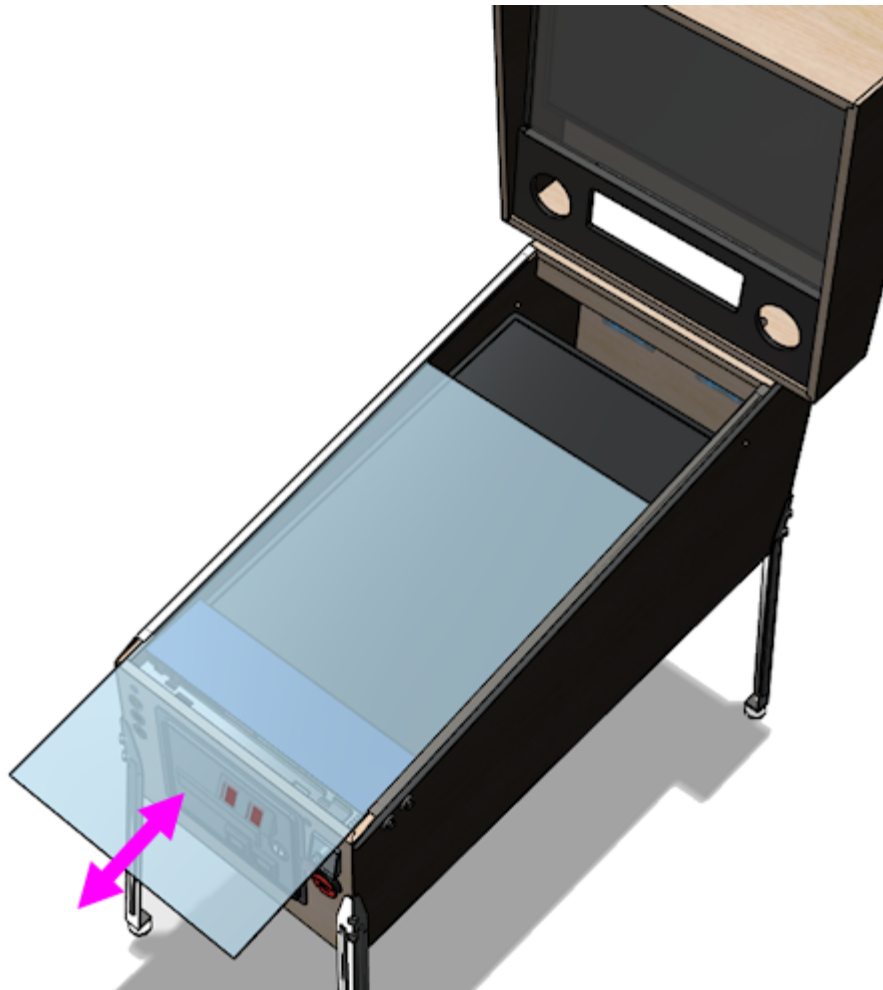
settings, but I think it's worth taking some precautions. One preventive measure that's often suggested is to use metal leg protectors, but I've seen mixed reviews of how well this works. The solution I prefer is to cut out the decals under the legs, to eliminate any contact between the legs and the decals. With the legs installed, take an X-Acto knife and cut through the decal around the perimeter of each leg. You can then peel off the part of the decal under the leg, or just leave it - even if that section wrinkles, it shouldn't affect the rest of the decal, since it's no longer attached.

Top glass

If you've set things up as we described above, with the plastic channels for the glass along the side rails and in back, the glass can be easily installed and removed at any time, without tools.

To install the glass, remove the lockbar, and slide the glass through the front of the machine, fitting the edges into the plastic channels under the side rails. Slide it back until it's nested in the trim at the back. Put the lockbar back in place to keep the glass from sliding back out on its own.

To remove the glass, simply reverse the procedure.



24. Backbox Hardware Installation

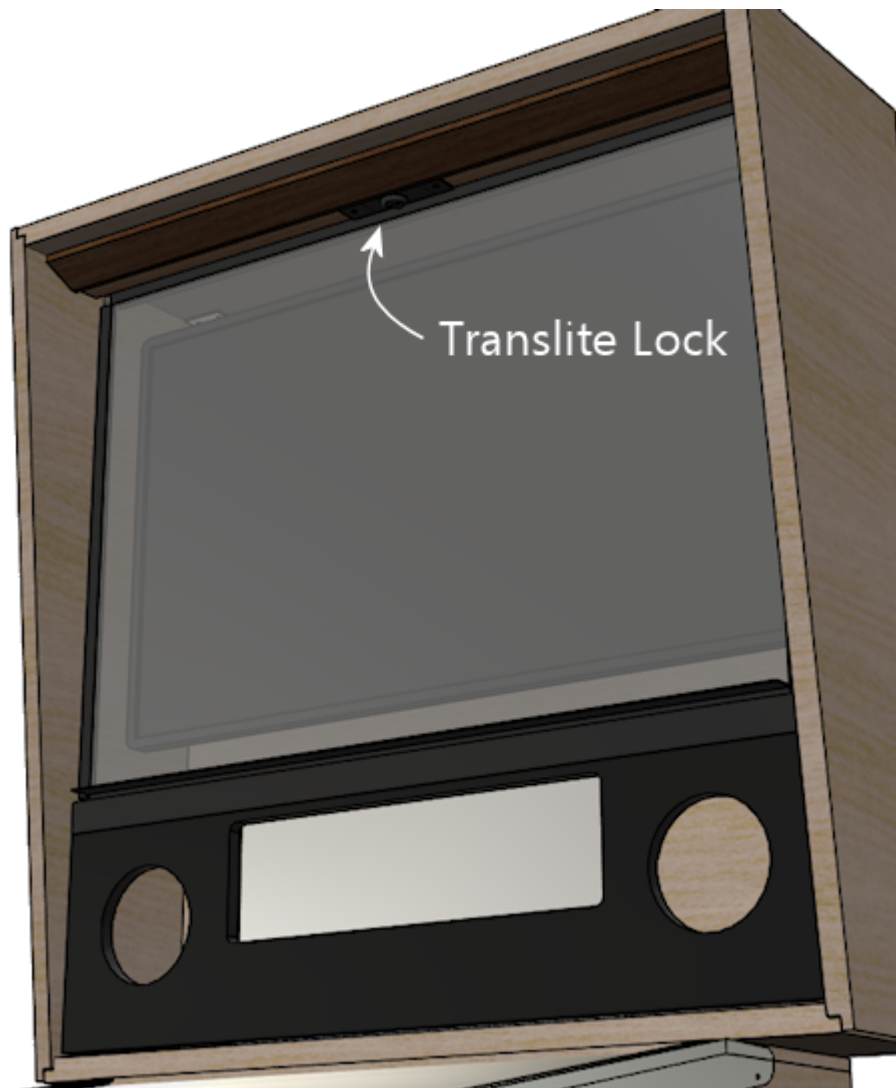
This section continues with the cabinet trim hardware, moving on now to the backbox.

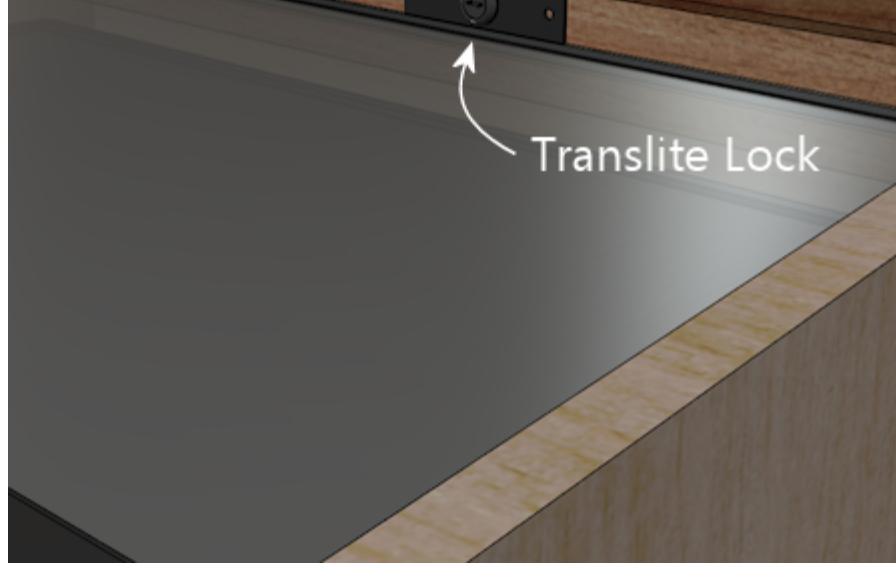
We assume that you've already built the wood shell of the backbox as described in Chapter 21, Cabinet Body, and that you've already painted it and/or applied decals, as described in Chapter 22, Cabinet Art. It's best to finish the artwork before installing any hardware, since some the hardware will get in the way of painting or applying decals once installed.

As with the Chapter 23, Cabinet Hardware Installation chapter, we try to present things in an order you can follow for installation.

Translite lock

The real machines have a keyed lock at the top of the backbox that secures the translite, so that arcade customers can't steal the translite or get into the backbox to mess with the electronics.





The operating principle is pretty simple. In the locked position, a metal tab on the lock sticks into the slot at the top of the backbox trim that the translite fits into. This prevents lifting the translite, which is necessary to remove it.



In the unlocked position, the metal tab swings out of the way, letting you lift the translite into the slot, which in turn lets you remove it.



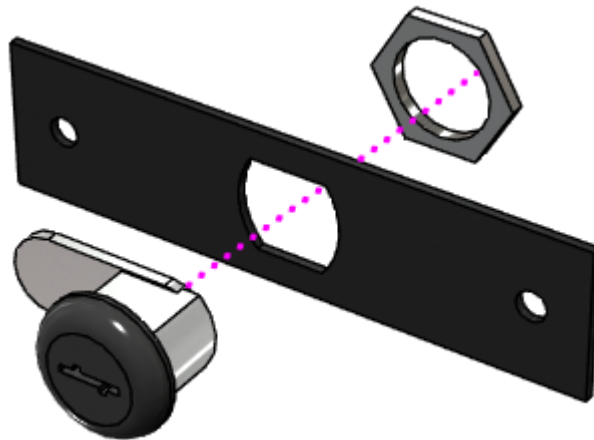


With the lock open, there's enough play that you can remove the translite, as described below. With the lock closed, the tab prevents moving the translite far enough to free it from the top and bottom trim channels, so it's effectively locked in place.

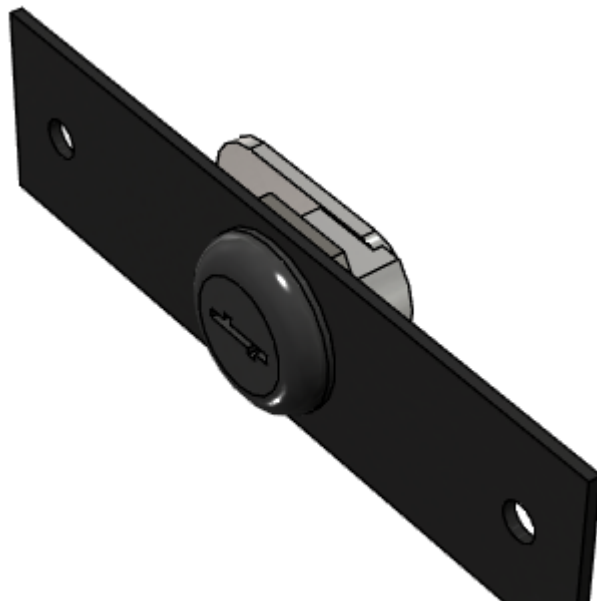
The translite lock is purely for the sake of security - it's there to prevent anyone without the key from removing the glass. The glass won't fall out on its own, though, even if you don't install the lock - it's held in place by the slots it sits in, and you have to intentionally maneuver it out of the slots to remove it. So it's not a functional necessity in a home setting, unless you have obnoxious friends. (The exception is that the glass could conceivably come loose during transport if you give it a bumpy enough ride. A lock does help prevent this by ensuring that the glass can't move out of its slots.)

Installing the translite lock

First, assemble the pieces of the lock plate. Slip the lock through the hole in the plate, slip the hex nut over the back of the lock, and tighten the nut.

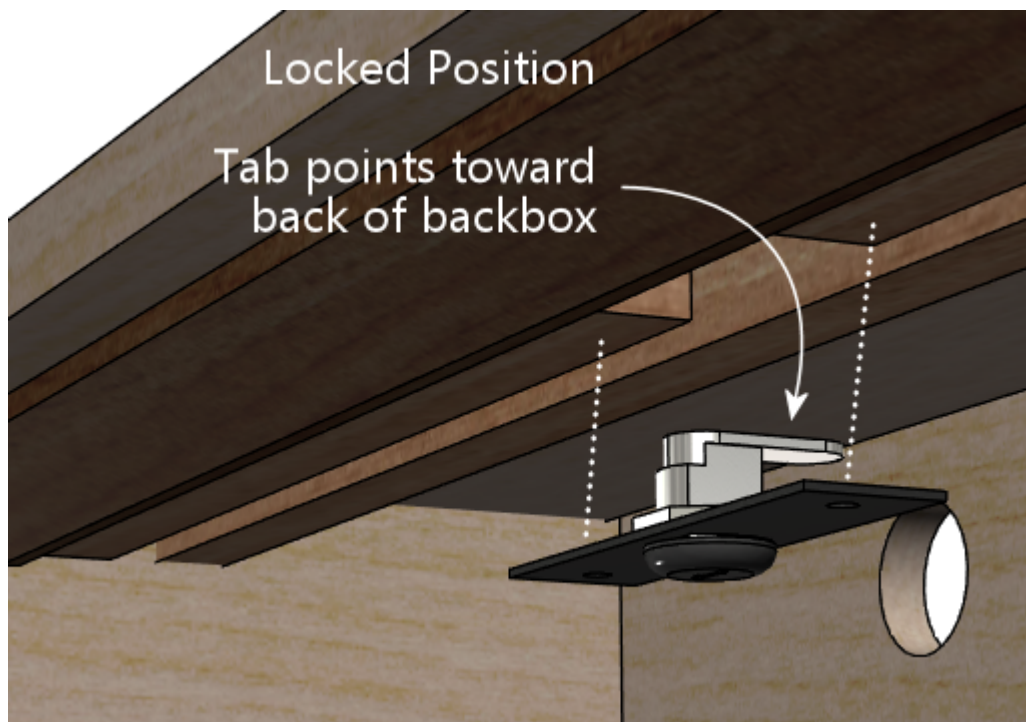


It should look like this when assembled.



The pinball vendors sell the lock plate assembly as a complete kit, which includes a pair #8-32 machine screws with security Torx heads. There are two reasons you might want to discard these and substitute your own wood screws. The first is that they're the security Torx type, so you need the special security type of Torx driver to use them. "Security screwdriver" is a bit hyperbolic when anyone can go buy one at Home Depot, but it's at least a slight deterrent against mischief simply because most people don't have one lying around. The second reason you might be inclined to discard the special screws is that they're machine screws, not wood screws. They won't attach well to plain wood. They require T-nuts, which must be pre-installed behind the trim, as explained in "Translite lock plate preparation" in Chapter 21, Cabinet Body. If you skipped that step when installing the trim, it's probably too late. Fortunately, wood screws are a pretty decent alternative, especially if you're not concerned about the security aspect of the lock. And if you are concerned about that, you could substitute tamper-resistant wood screws.

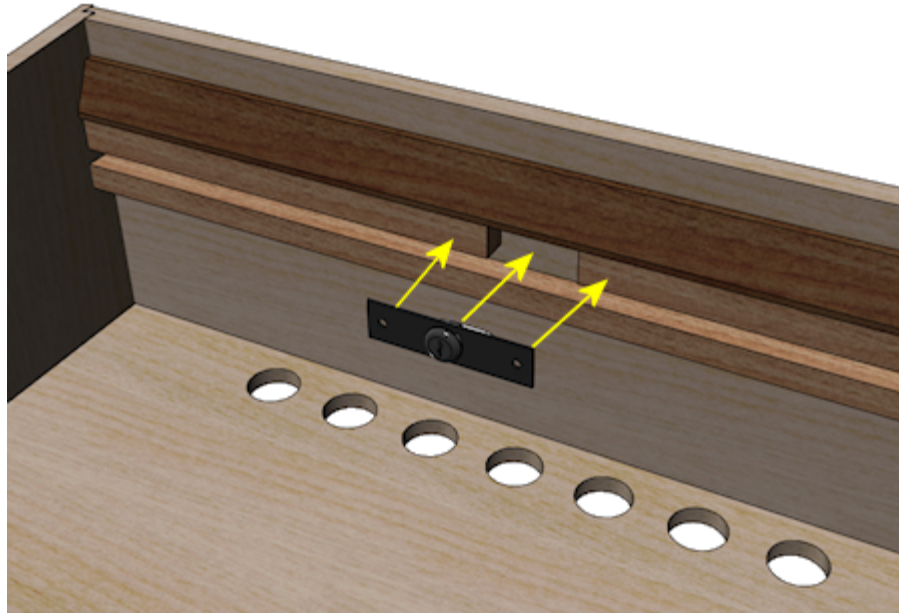
Before installing the lock plate, use a key to check the orientation of the lock. Turn the lock so that it's in the extended position, with the lock tab sticking up perpendicular to the plate. Be sure to install it with the tab facing the **back** of the backbox.



If you did already install T-nuts for the Torx screws, simply put the lock plate in position, and fasten it with the Torx screws.

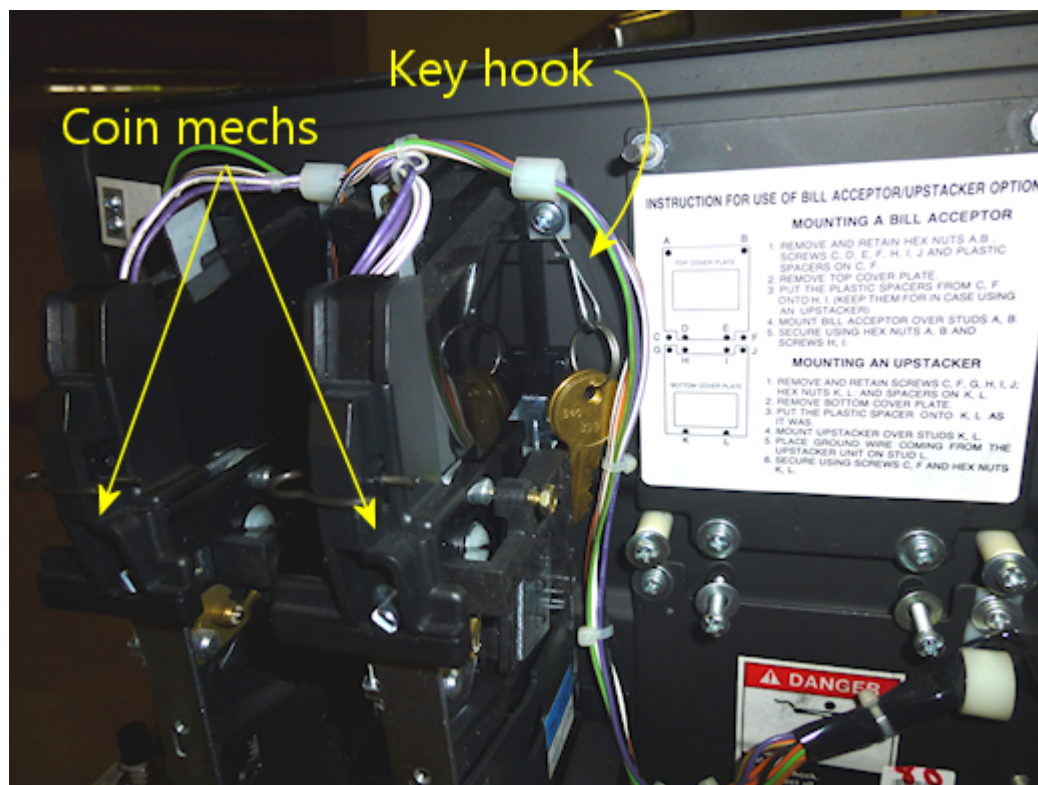
If you didn't install the T-nuts, discard the Torx screws that came with the kit, and substitute a pair of wood screws. I'd go with #6 x 3/4". Rounded-head screws will look better, as will black screws if you can find them (if not, you can just paint the heads black after installation if you want).

Line up the lock plate over the gap in the top trim. It should be centered over the gap, which should be the same as centering it side-to-side overall in the backbox. Mark the positions of the screw holes. Remove the plate and drill pilot holes for the #6 screws. Put the lock plate in position again and fasten the screws.



Where to keep the keys

With the real machines, the standard way to keep track of your translite keys is to keep them on a little hook inside the coin door. The WPC and SuzoHapp doors provide a hook specifically for this purpose, located next to the coin mechs.



If you're not installing a standard coin door or can't find the key hook, you might put a little eyelet or hook on the inside wall of the cab somewhere convenient, and hang the keys there.

DIY alternatives

If you don't need the locking function, but your backbox has the gap in the trim where the lock plate goes, I'd simply install a 1" x 4" plate, either metal or a thin piece of wood, painted black. Screw it in with #6 x 3/4" wood screws, at holes placed about 3/8" from either edge. Use rounded-head screws, and either use black screws or

paint the heads black after installation.

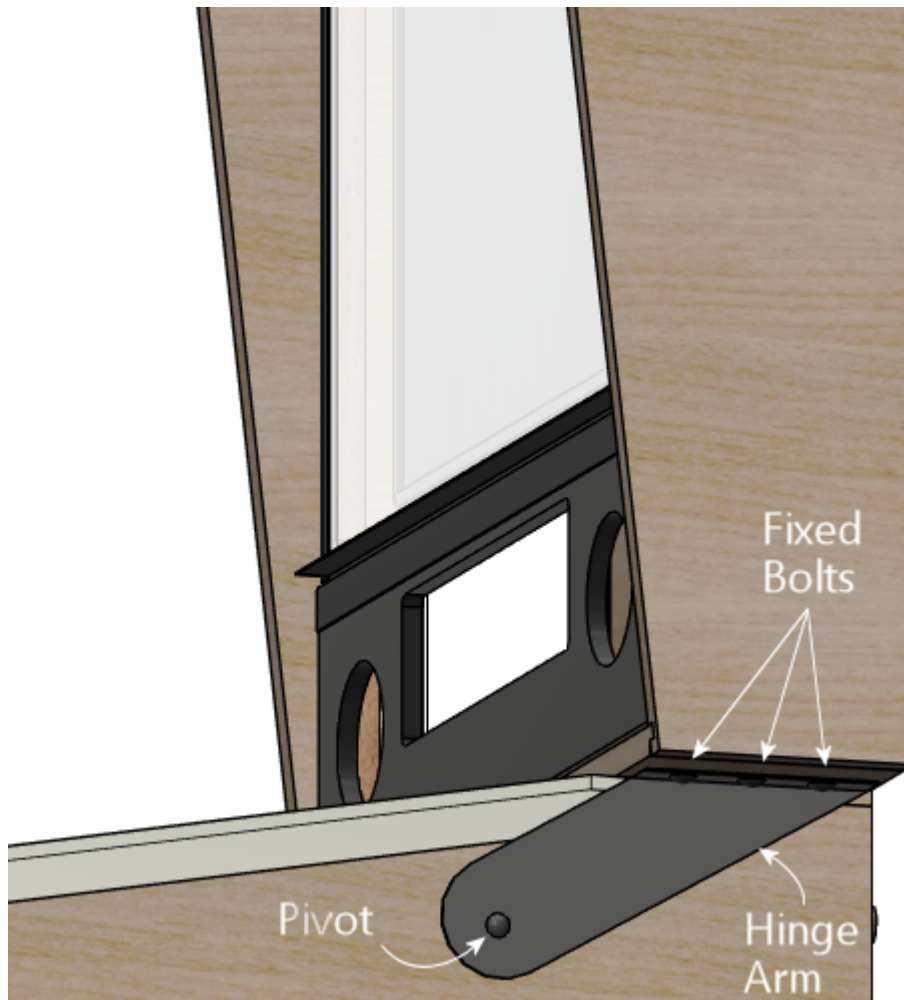
If you haven't yet installed the wood trim where the plate goes, I'd simply run a single piece of wood all the way across rather than replicating the gap in the standard plans.

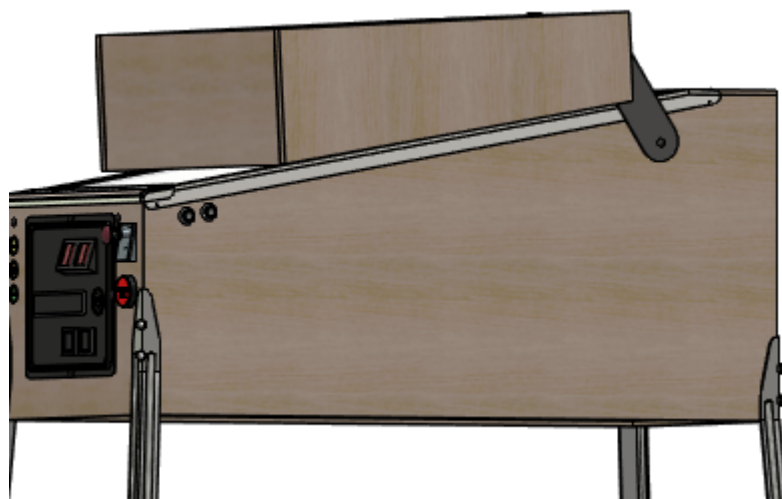
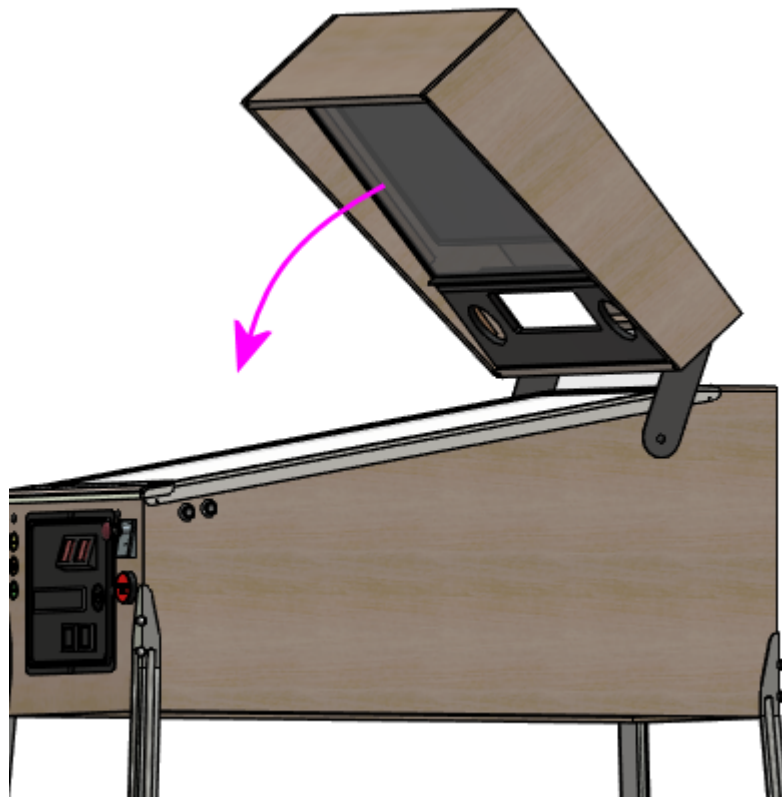
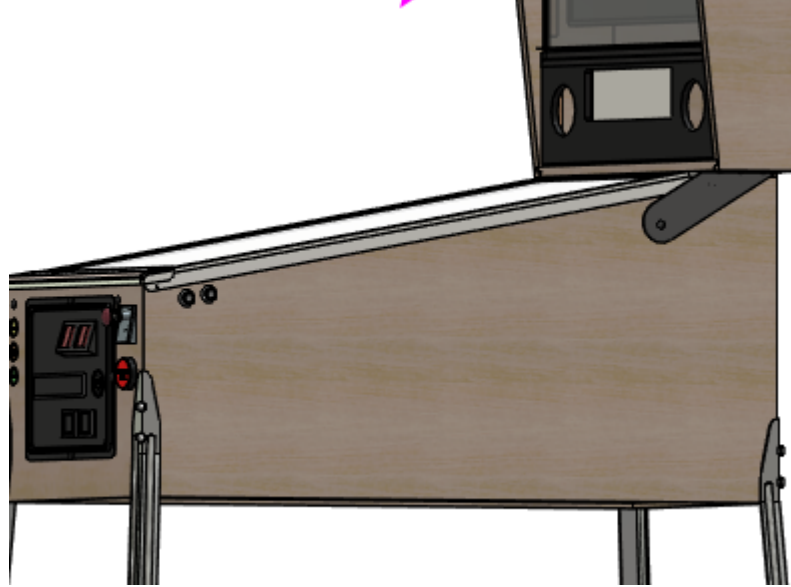
Backbox hinges

Real pinball machines are designed with a hinge that lets you tilt the backbox forward until it's lying flat on top of the main cabinet. This is an essential feature if you want to transport a pinball machine, as it would be too tall, top-heavy, and fragile with the backbox in the normal position. And you certainly wouldn't want to remove the backbox every time you moved the machine, as that's a major operation that requires disconnecting a lot of wiring, which always creates a risk of breaking something when you reassemble it.

WPC hinges

On the WPC machines, the backbox is attached to the main cabinet with a clever hinge mechanism that uses rotating arms connected to pivots on the sides of the cabinet.





I consider this setup clever, in part because it wouldn't have occurred to me to do it

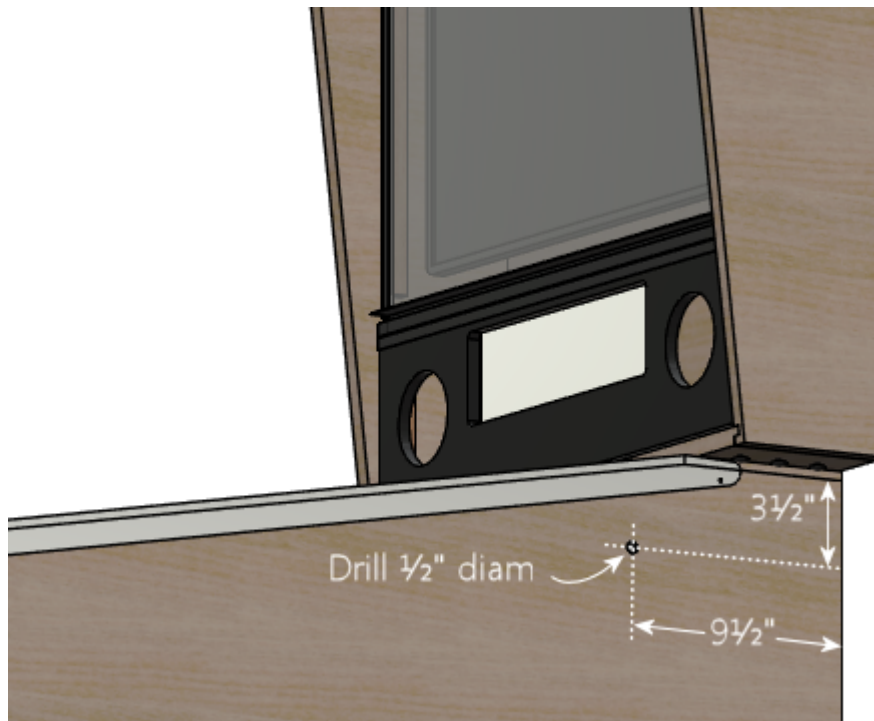
this way, but mostly because it has some nice functional advantages over the more obvious "door-hinge" approach, where you simply put a regular hinge at the front of the backbox. A door-hinge approach is workable, and in fact it's used on a lot of older pinball machines from before the WPC era - we'll have more on this shortly, since it's a viable alternative if the WPC approach doesn't work for you for some reason. But the WPC system more elegant: it looks nicer, it makes for simpler cabinet geometry, and it's easier to install and to get everything aligned properly. Door-hinge systems can be tricky to get aligned.

So assuming you're building a full-scale cabinet, the WPC approach is all upside. And it's not particularly expensive; the required parts will only set you back about \$30.

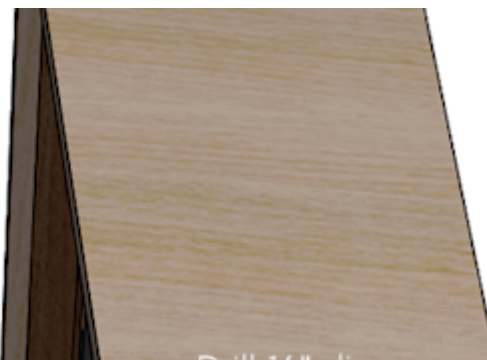
The only reasons I can think of not to use the WPC hinges are (a) that you're building a mini-cabinet that's too small for the standard parts to fit properly, or (b) that you're specifically reproducing a cabinet style from a different era, so you want to use the same hinge system they used. If one of these applies to your build, you can skip down to the "Alternatives" section below for some other ideas.

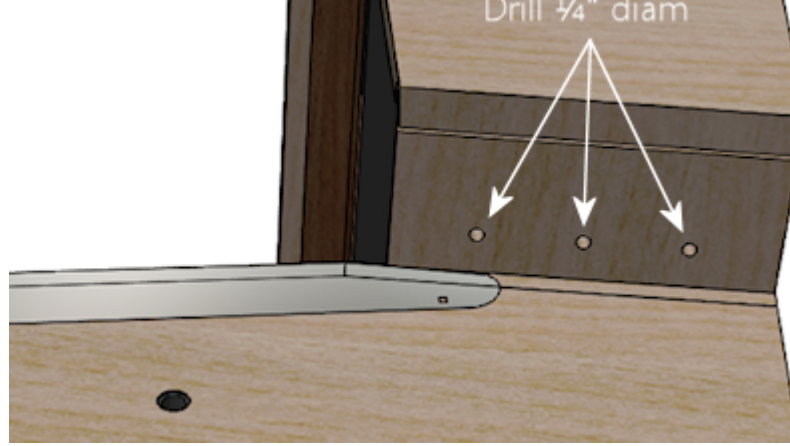
Installing the WPC hinges

One of the advantages of the WPC hinge mechanism is that it's easy to set up. If you followed the WPC cabinet plans in Chapter 21, Cabinet Body, you've already drilled the mounting holes in the main cabinet and backbox. To review, the parts get mounted here:



...and here:





See "Backbox floor" in Chapter 21, Cabinet Body for drilling locations, which depend on your cabinet width. If you haven't already drilled these, an easy way to figure the exact position is to use the hinge arm itself as a drilling template, after attaching it to the main cabinet first. We'll point out the right time to do that in a moment.

You should already have your side rails installed at this point, since the hinges will get in the way of installing them later.

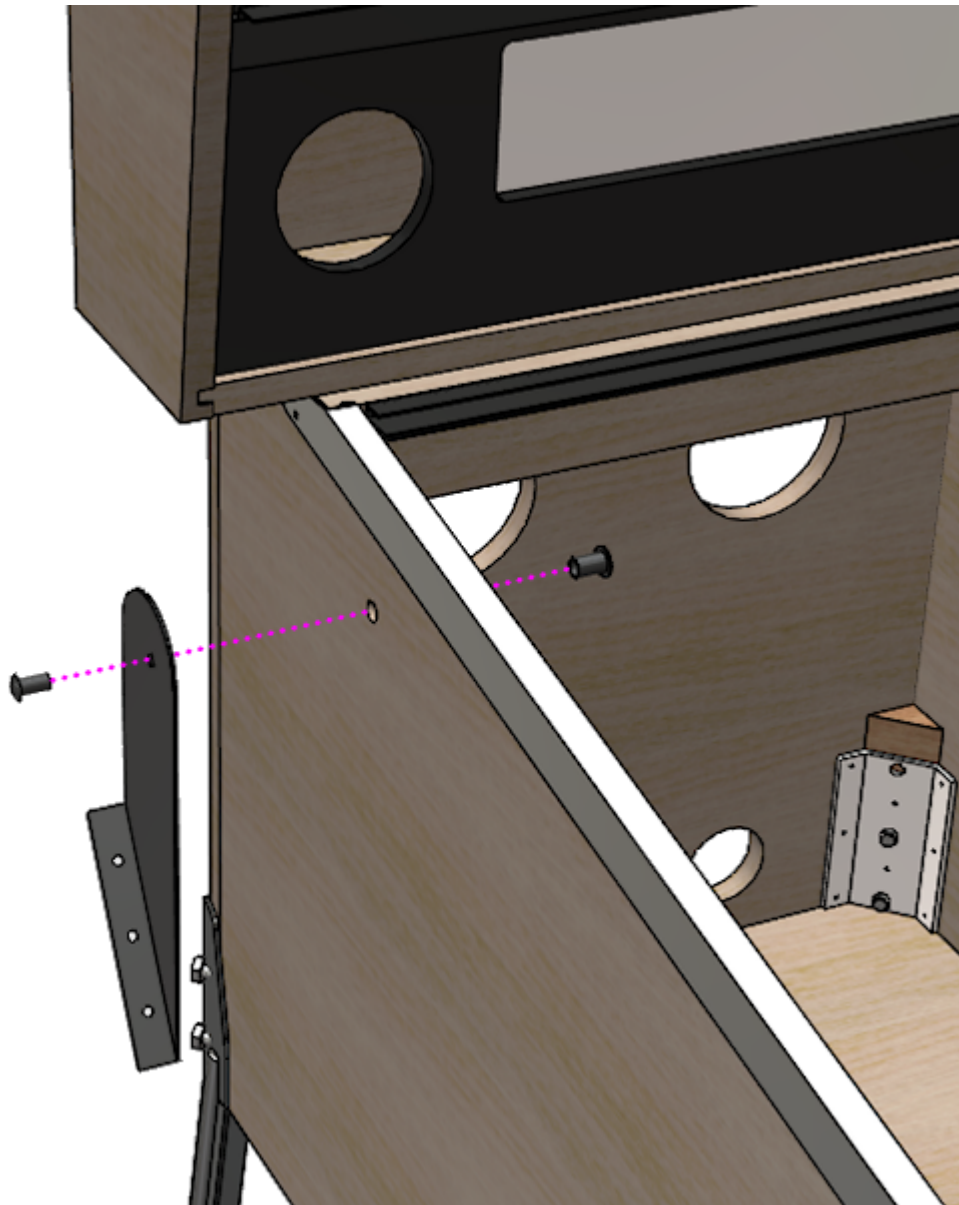
Note that there's a "left" and a "right" hinge arm - they're mirror images. Be sure to install the correct one on each side. They should be oriented with the pivot point pointing towards the front of the cabinet, and the little "wing" that attaches to the backbox pointing away from the cabinet side:



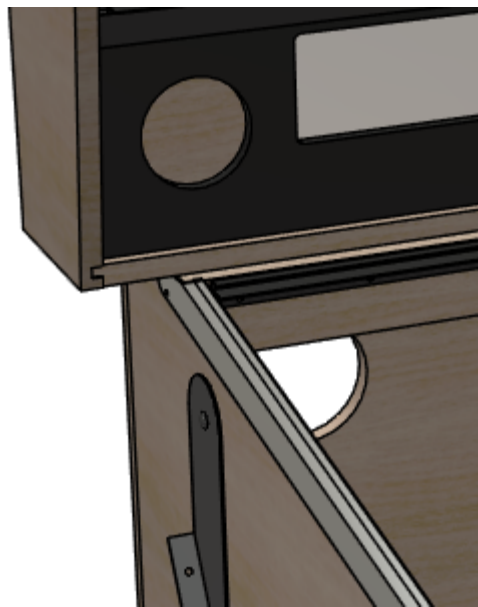
Start with the hinge pivot joint. This uses the $\frac{1}{2}$ " diameter hole drilled in the side of the main cabinet:

- Fit a $\frac{3}{8}$ "-16 x $\frac{3}{4}$ " carriage bolt into the squarish opening on the side of the hinge arm. Orient the hinge arm so that it's hanging from the bolt, so that it doesn't swing down into this position on its own and scratch the side of the cab.
- Insert the carriage bolt into the pivot hole in the main cabinet
- From the **inside** of the cabinet, thread the pivot bushing (Williams part 02-4352) by hand onto the end of the carriage bolt

- Use a 1/4" hex wrench to tighten the pivot bushing from the inside of the cab



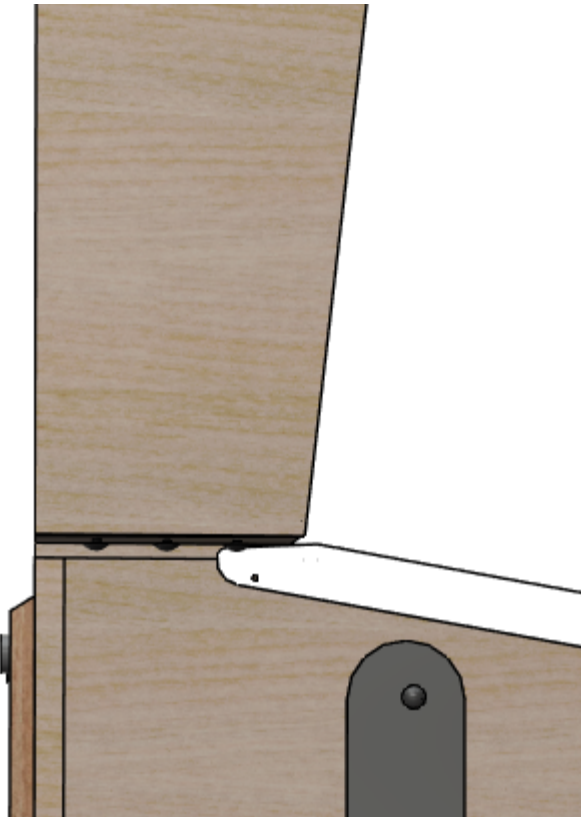
When tightened, this arrangement should leave a little clearance (about 1/8") between the hinge arm and the main cabinet, and you should be able to rotate the hinge arm around the pivot. (It's okay if it's tight.)



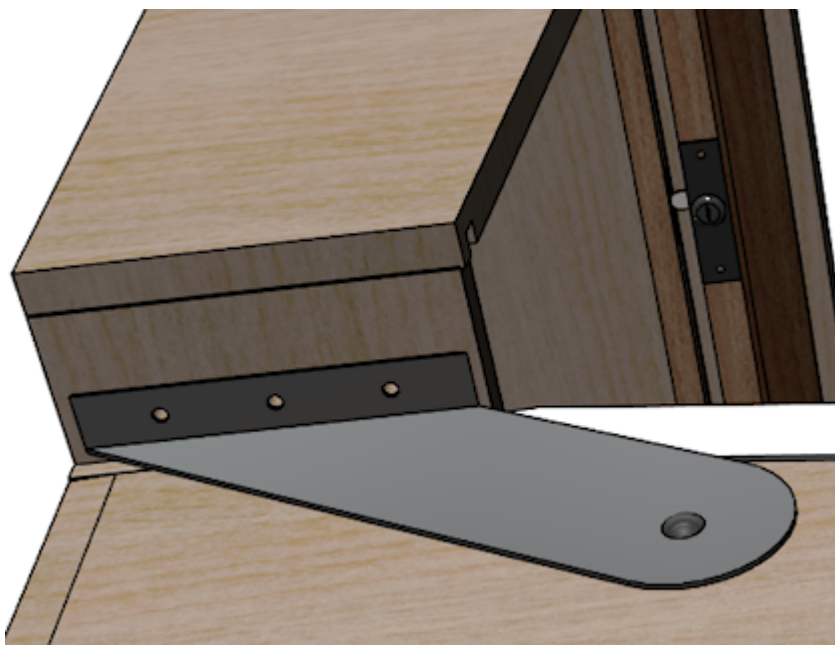


Install both hinge arms (left and right) using the procedure above.

Position the backbox on the shelf at the back of the cabinet, centered side to side, with its back flush with the back wall of the main cabinet. (Have an assistant hold the backbox steady while you're working so that you don't accidentally knock it over.)



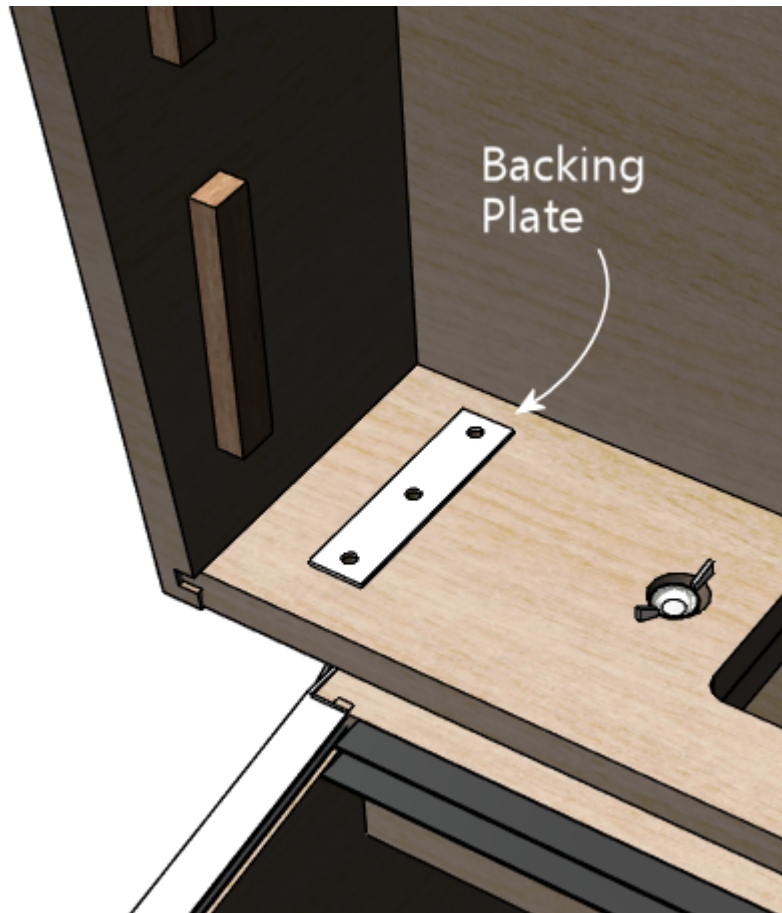
Rotate the hinge arm around the pivot until the side with the three bolt holes meets the bottom side of the backbox. Be careful about rubbing the sides of the cab so that you don't scratch the artwork.



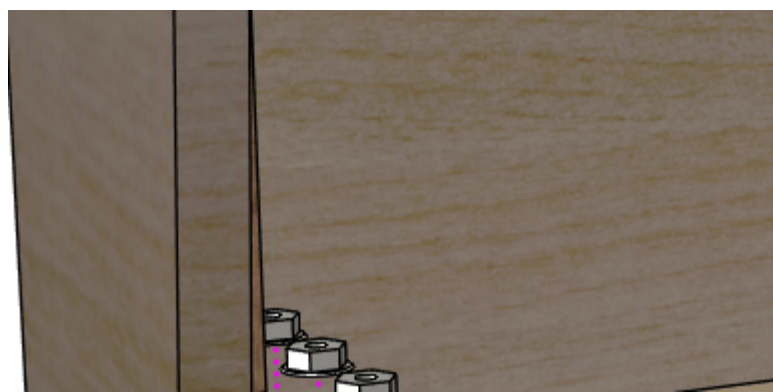


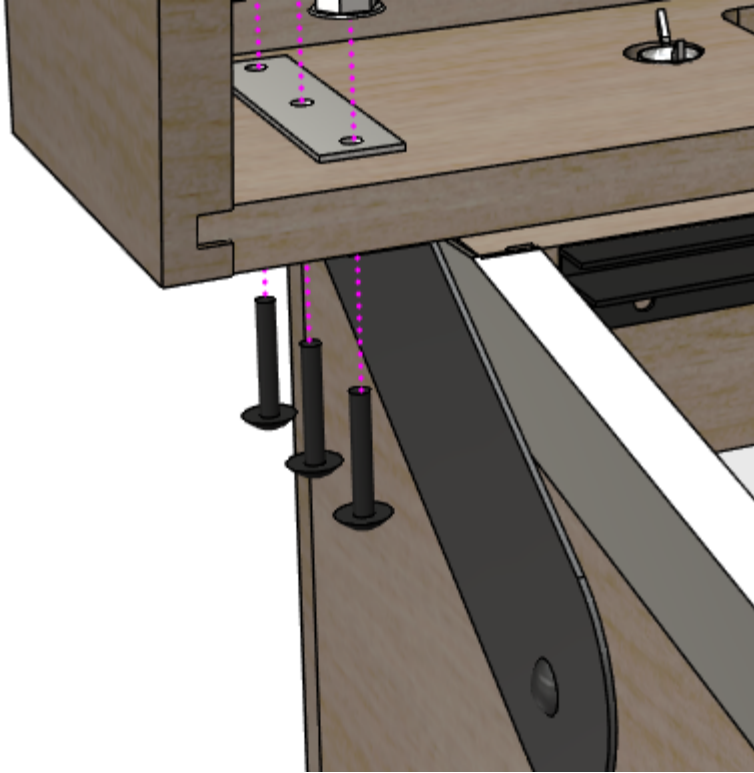
If you haven't already drilled the holes in the backbox floor for attaching the hinge bracket bolts, this is the time! Make sure the hinge arm is flat against the bottom of the backbox, and that it's precisely parallel to the side of the cab. You should be able to see a little gap between the hinge arm and cab across its whole length. The hinge shouldn't be pressing against the cab anywhere, since that could scratch the artwork when you rotate the backbox. Once you have it aligned to your satisfaction, mark the positions of the three bolt holes. Repeat on the other side. Remove the backbox and drill $\frac{1}{4}$ " holes at the marked positions.

On the inside of the backbox, position the backing plate (01-9012) over the bolt holes.



Install three $\frac{1}{4}$ "-20 x $1\frac{1}{4}$ " carriage bolts in each hinge arm, inserting from the bottom side, and through the mounting plate. Fasten each with a $\frac{1}{4}$ "-20 whiz flange locknut. Tighten securely.





The backbox is now attached! You should be able to freely tilt it forward so that it lies flat against the top of the cab. (It's a good idea to put down some padding when doing this, so that you don't scratch up the side rails or the front edges of the backbox.)

If you ever need to remove the backbox, just take out the carriage bolts attaching the hinge arms to the bottom of the backbox. You can leave the hinge arms themselves attached permanently.

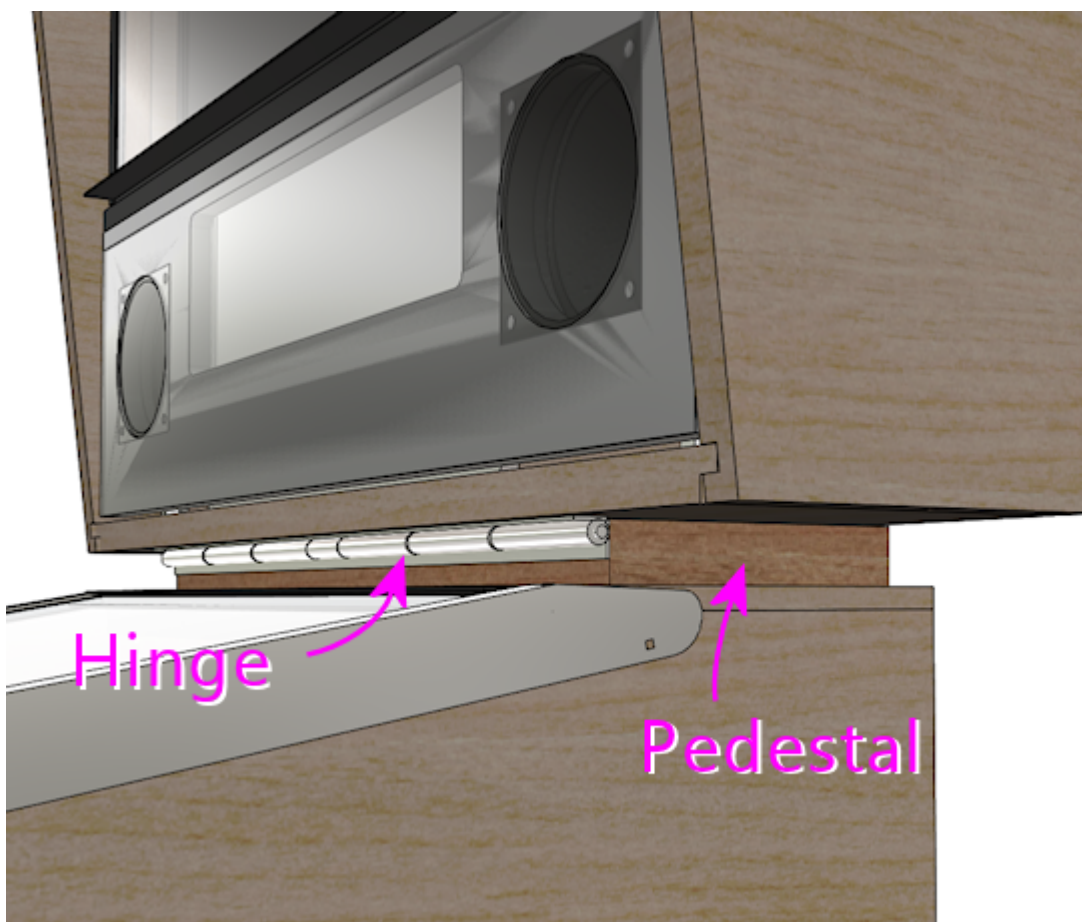
Alternative hinge mechanisms

Most commercial machines made from the 1990s to present use the WPC hinge system described above. Most earlier machines that I've encountered use something more like conventional door hinges, with the hinges attached at the bottom front of the backbox. The backbox folds down forwards onto the cabinet, as in the WPC system, but the pivot point is the door hinge rather than the side bolts. The Williams System 11 machines from the 1980s use this approach, as shown below.





Williams System 11 backbox mounting (Space Station, 1987). This used door hinges at the front of the backbox. Note how the backbox has to be raised slightly above the cabinet on a pedestal, to make space when folded down for the part of the backbox that overhangs the front of the hinge.

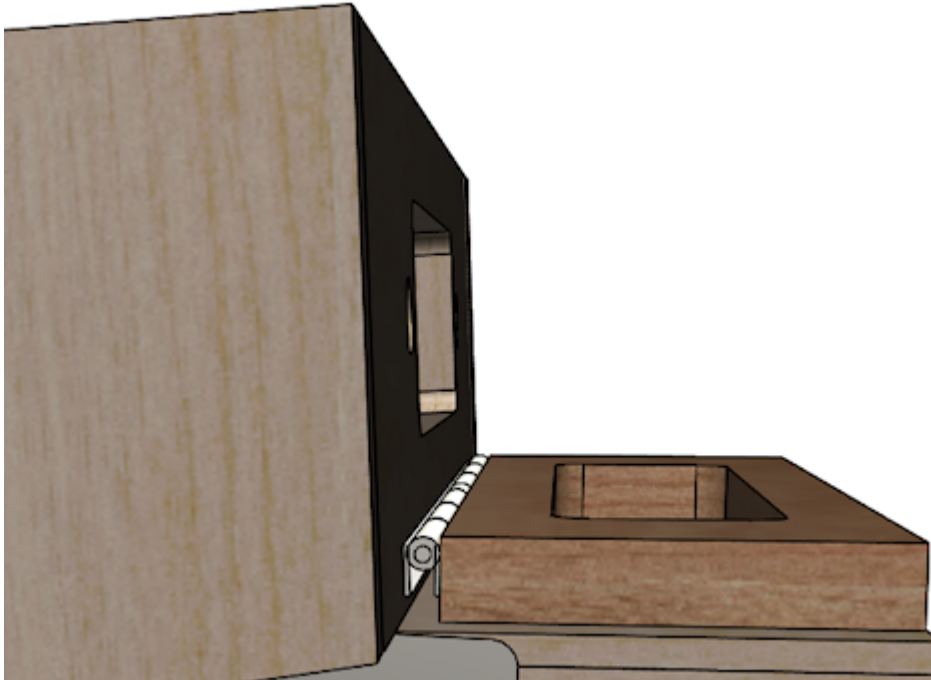


I think the WPC system is nicer in a lot of ways, but the door-hinge system might be a good alternative in cases when the WPC parts won't fit, such as a mini-cab or an ultra-wide cab. If you use the System 11 machines as your model, pay close attention to the way it requires a "pedestal" to raise the backbox about an inch above the main cabinet, to make room for the front overhanging portion of the backbox when it folds down. The standard WPC plans won't work well with a hinge like this, because the backbox sits directly on top of the cabinet in the WPC design. If you want to adapt the WPC plans for this arrangement, you'll have to add something like the System 11 pedestal.





The pedestal has to be at least as high as the distance the backbox projects out in front of the hinge, to make room for that part when the backbox is folded down.



Hinge system with backbox folded down.

Backbox latch

This is a minor bit of hardware that helps when setting up the machine, by temporarily securing the backbox in the upright position, preventing it from falling forward if bumped. The standard part is a simple toggle latch that attaches to the back of the main cabinet, with a mating bracket that attaches to the back of the backbox.

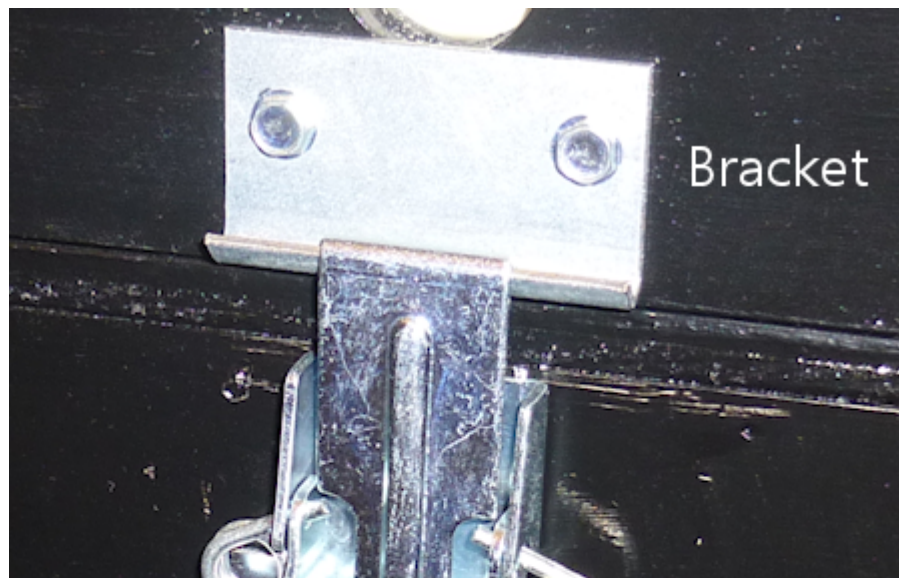
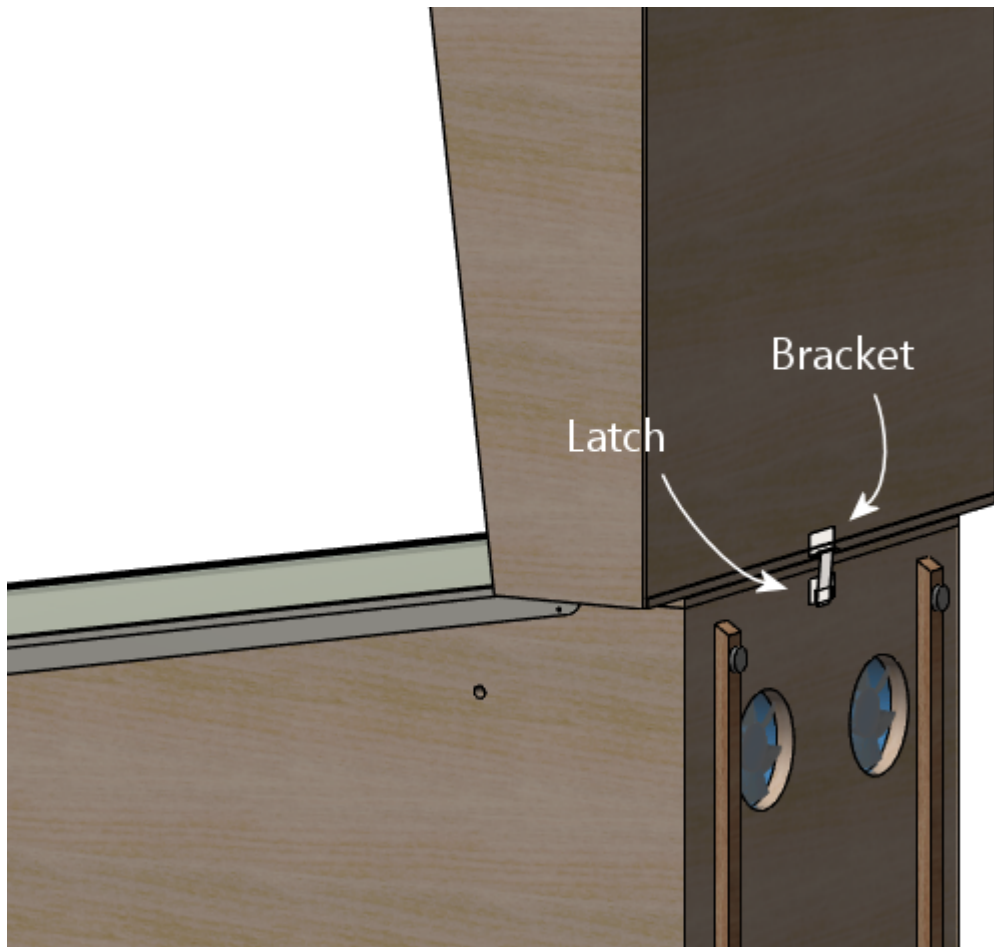
I said "temporarily", because the toggle latch isn't strong enough to serve as a permanent way of securing the backbox. Let me show you the warning that they silkscreen on the back of the real backboxes in big yellow letters:

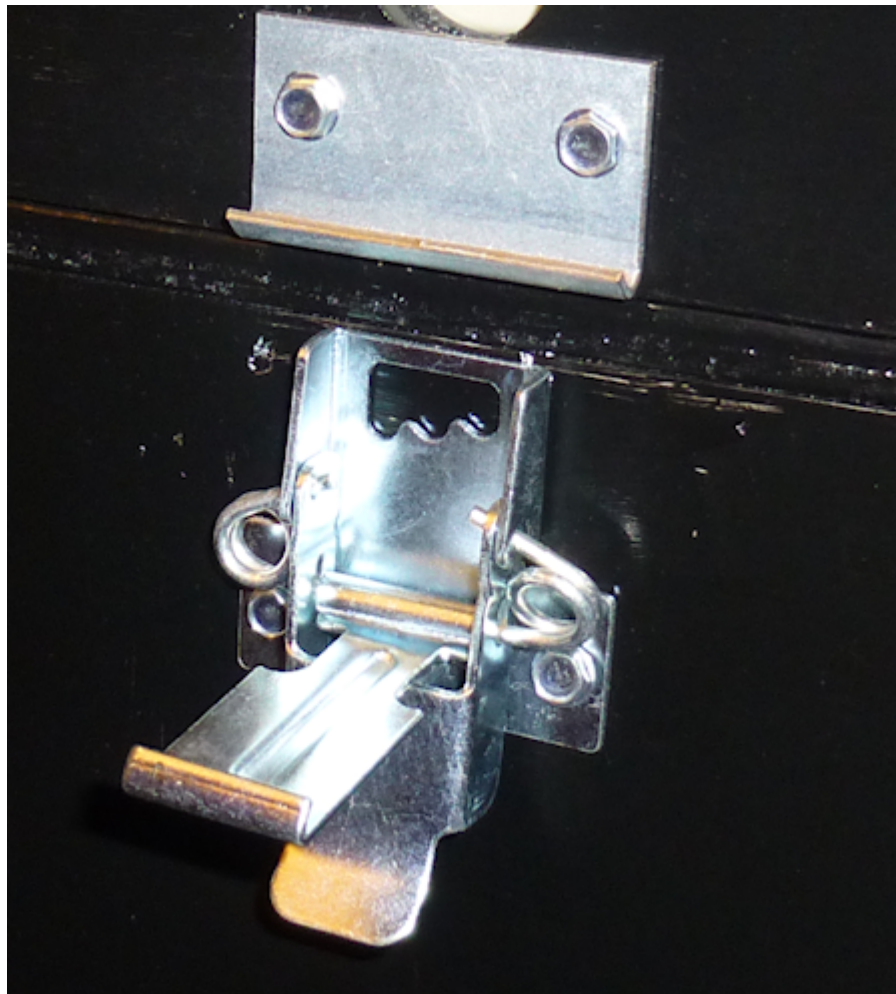


BACKBOX BOLTS ARE TO BE INSTALLED AT ALL TIMES.

FAILURE TO DO THIS MAY RESULT IN POSSIBLE PERSONAL INJURY OR DAMAGE TO THE BACKBOX AND CABINET.

Their point is that this little toggle latch isn't all that strong; it could fail if the backbox were bumped too hard. The backbox is quite heavy and has a lot of leverage, so you need something a lot stronger to truly secure it. The solution is to install the wing bolts described below. The toggle latch is just meant to be a temporary helper while you're getting the wing bolts in place.





Install this after you've set up the backbox hinges, so that you're working in terms of the actual final alignments.

For fasteners, use any suitable wood screw. On the real machines, they usually use #6 x $\frac{3}{4}$ " sheet metal screws with hex heads. (I know, "sheet metal screw" doesn't sound like the right thing for screwing into wood, but they actually work just fine as self-tapping screws with plywood.)

- Set up the machine with the backbox in the upright position. Have an assistant brace the backbox while you're working so that it doesn't fall forward.
- Attach the bracket (the top piece) first. Align it in the center of the backbox side-to-side, with the bottom edge roughly flush with the bottom of the backbox.
- Figure the position of the latch itself by hanging it from the bracket with the lever pulled partially open, so that there's no tension on the spring, as

illustrated below.



- Fasten the bottom bracket at this position. When you close the latch all the way, it should pull the spring tight so that the bracket stays latched.

Backbox safety bolts

As the warning placard above points out, the little toggle latch on the back of the backbox is helpful to keep the backbox from flopping over while you're setting up the machine, but it's not strong enough to rely on beyond that. For a deployed machine, you need something stronger, specifically a couple of big bolts installed in the floor of the backbox.

The parts required are a pair of $\frac{3}{8}$ "-16 x 2" "wing bolts", like the one pictured below. Wing bolts are basically just regular bolts with wing nuts in place of the heads. This lets you turn them by hand, for tool-free installation. (The wing nut at the top isn't a separate part; it's integral, like the hex head on a regular bolt.) The wing nut head also serves as a built-in washer.





You can buy wing bolts in the right size from pinball vendors. You might be able to find them at some hardware stores as well, although they're too obscure for the big-box stores like Home Depot. In a pinch, you could substitute ordinary hex-head bolts of the same size, but note that you'd need to use some kind of giant washers (over 1" outside diameter) in conjunction, because the hex heads by themselves will slip right through the 1" top holes (defeating the purpose).

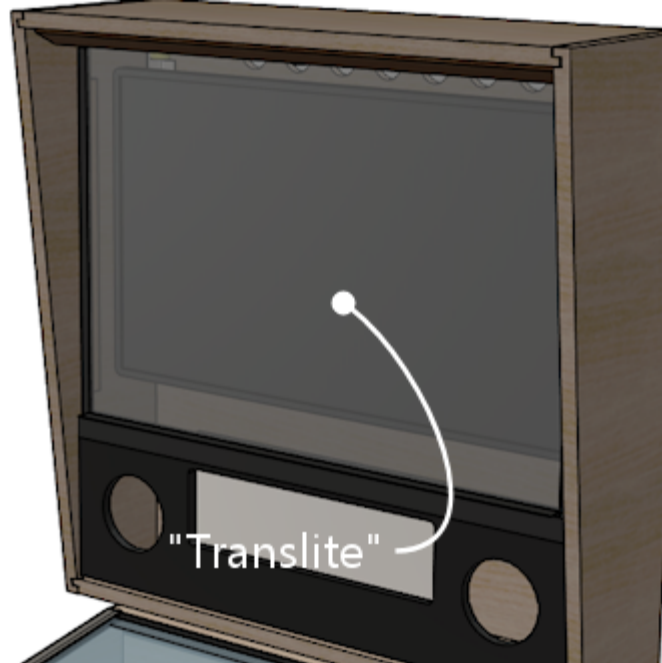
If you've done the necessary prep work as described in the "Rear shelf" section in Chapter 21, Cabinet Body, installing the bolts is trivial. Just pass them through the holes on either side of the floor opening and thread them into the pre-installed T-nuts. There's no need for washers or other parts. Hand-tighten. You don't have to go beyond hand-tight, since they don't have to hold the backbox up most of the time; gravity takes care of that for the most part. The bolts only kick in if the backbox gets bumped or pushed.



What is a translite?

We're all pinball nerds here, so a quick digression on definitions is in order! There's a bit of a disagreement between virtual pinball people and real pinball people about what "translite" means. So to avoid confusion, I want to make sure we're all clear on what we mean by the word.

Throughout this guide, I use the term "translite" to refer to *a clear plastic or glass sheet that you install in front of the main backbox TV*. There's no artwork printed on it (other than perhaps some masking around the edges, to hide the TV bezel), since we want to let the TV handle all of the artwork display. The virtual translite is thus essentially a bit of trim in a virtual cab to disguise the TV-ness of our virtual setup and make it look on the outside more like a real pinball machine.



Okay, so that's how *I* use the term in this guide. It's also how the term is commonly understood in the virtual pin cab community, so that's the way you'll usually see it used on the forums. But technically speaking, it's wrong! At least, it's not what it means to (most) pinball people when they're talking about the real machines. Technically, in a real pinball context:

- A *translite* is a thin, translucent plastic decal, printed with graphics. There's no glass involved in the translite itself. This plastic decal that they call the translite is then affixed to a clear glass sheet to create a pseudo-backglass. Most modern pinball machines (1990s and later) use this type of assembly in place of a true backglass. It's only a "pseudo" backglass because...
- A *backglass* is a glass sheet with artwork directly painted or silkscreened on the glass. True backglasses were usually used on machines built before about 1990, when the manufacturers switched to the cheaper translite process.

Those are the technical meanings, but even real pinball people often use the words *translite* and *backglass* loosely and interchangeably. They'll often call the translite-plus-glass assembly a translite, or even a backglass. So I think we virtual pinball people can be forgiven for appropriating *translite* to mean kind of the opposite of what it really means, in that we use it to refer to the plain glass sheet without any artwork.

Creating a translite

The basic material for the translite is a clear sheet of either glass or acrylic. If you're using glass, it should be tempered glass. I personally prefer acrylic for the translite because it's so much lighter than glass.

Dimensions for the standard WPC-style translite:

- Thickness: $\frac{1}{8}$ "
- Size: $18\frac{7}{8}$ " high x 27" wide

The size obviously depends on your backbox dimensions. The size above is for the standard WPC setup, with the backbox built to the standard dimensions and a standard-sized speaker/DMD panel installed. If you're not using a speaker panel, you'll probably want to increase the height to cover the entire backbox area, so you'd

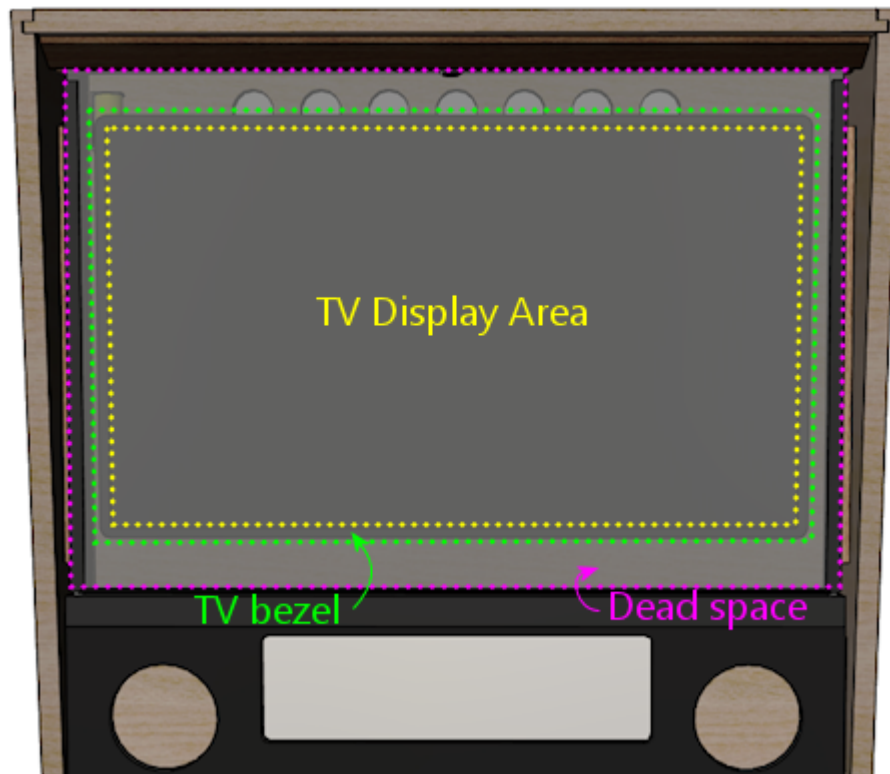
make it about 26½" high if you're using the standard backbox dimensions.

Where to buy: The clear glass or plastic sheet isn't something you can find as a standard pinball part from any vendor. Fortunately, it's easily found as a generic part.

- For glass, you can have a custom glass sheet made by just about any window glass company. Look for local companies that install or repair residential window glass.
- For acrylic, try TAP Plastics or a similar local plastics vendor. You can also buy acrylic in standard sizes at Home Depot and other hardware stores, and cut it to a custom size yourself using a plastic knife. This doesn't tend to make as clean an edge as you'd get from a plastics shop, but that doesn't really matter, since the edges are all covered by trim pieces anyway.

Edge masking

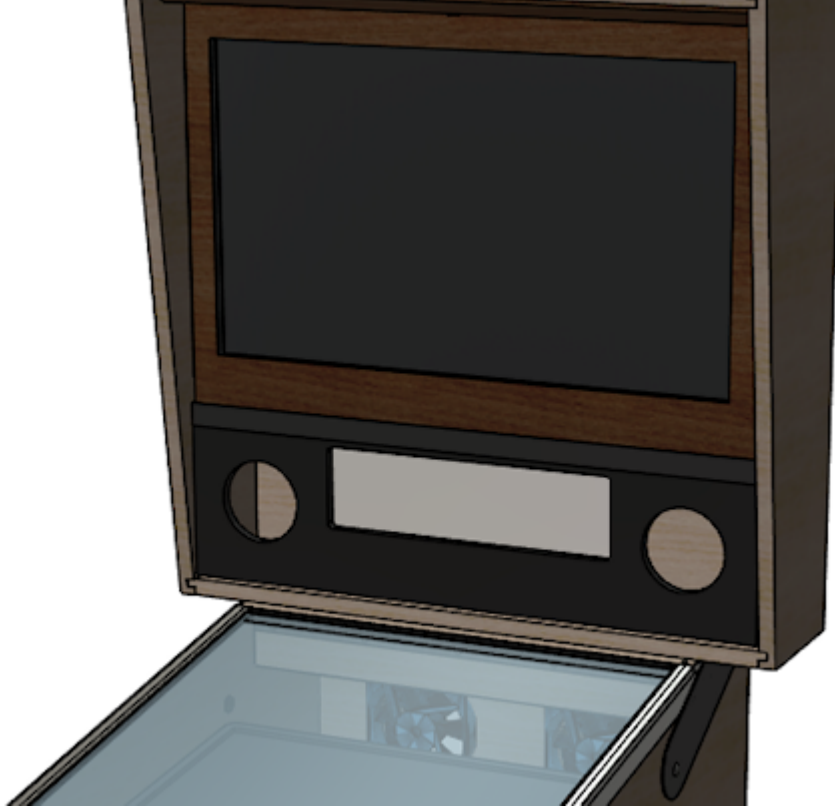
It's nearly impossible to make the backglass TV fill the entire space that's available for the translite in the standard backbox design. Part of the reason is that a TV's viewable screen area never completely spans the full height and breadth of the unit; there's always at least a thin bezel around the edges. The other factor is that the backbox space is much squarer than the 16:9 aspect ratio that's all but universal on current TVs. It's impossible to find a TV that fills the vertical space fully, given the constraint of also fitting in the available width. You can get down to a fraction of an inch of dead space on the sides, but you'll still have more than an inch at the top and bottom in the best case.



Most cab builders want to cover up all of the dead space around the edges of the TV, so that only the live display area of the TV is visible. Understandably, they don't want the insides of the backbox to be visible.

One way that some cab builders deal with this is to create a wood (or similar) cover with a cutout for the TV area.





I don't personally like this look very much, because to my eye, it calls attention what it's meant to hide. I mean that it makes it more even more obvious than it otherwise would be that there's a smaller TV embedded in a bigger backbox space. I also find that it looks too different from the real machines, which creates an impression of home-brew-ness that's at odds with my goal of a realistic appearance.

Given that we're talking about translites, I think you can probably guess that the approach I prefer is to use a clear glass or plastic cover for the whole area. That's exactly what the real machines use, so it looks as close to authentic as you can get, given the inherent differences in what's behind the glass. But that still leaves the problem that you can see into the dead space around the TV, since a clear glass or plastic cover is, after all, clear.



The best solution, in my opinion, is to combine the "cutout" idea from the wood cover with the clear translite, by masking out the edges of the translite. There are two ways to do this:

- The easy way is to paint around the perimeter with black spray paint. Paint on the back side of the panel, so that the front has a uniform glossy sheen - that'll largely eliminate the visibility of the "cutout" that I find objectionable in the wood cover approach. Measure the TV display area size, and use masking tape and paper to cover the cutout area. Paint around the edges.
- The other way is to use printed decals to create the mask. You can have custom decals made for this use just like for the cabinet artwork (see Chapter 22, Cabinet Art). I used this approach for my own cab, because I figured the real machines have artwork here, so I should too.



I used decals printed in the conventional way, to adhere to the front side of the translite. It would have been better to print them in a reverse format, with the adhesive on the graphics side so that they could have been stuck to the back side of the translite. This would have created a more uniform finish on the front, just like why you want to paint on the back if you're using a painted mask.

I really like the way my translite with decals turned out, but for practical purposes you might be better off with a simple black mask. The thing that you might not expect about the decals is that you simply don't see them while playing. The TV display is so much brighter that it completely overwhelms them to the point of utter invisibility. Which is exactly what you want, as it turns out: you want it to look like the backbox of the game you're playing, not like a TV embedded in a virtual cab. So that works out great, but my point here is that as far as playing goes, there's no difference between decals and black paint. And when the cab is powered down, the decals arguably create the same visual impression that I said I *don't* like with the wood cutout approach - the way the visible borders call attention to what's missing in the middle. But somehow I don't dislike the look in this flat "2D" version; in person, it actually looks very much like a real translite, even if it would be a rather

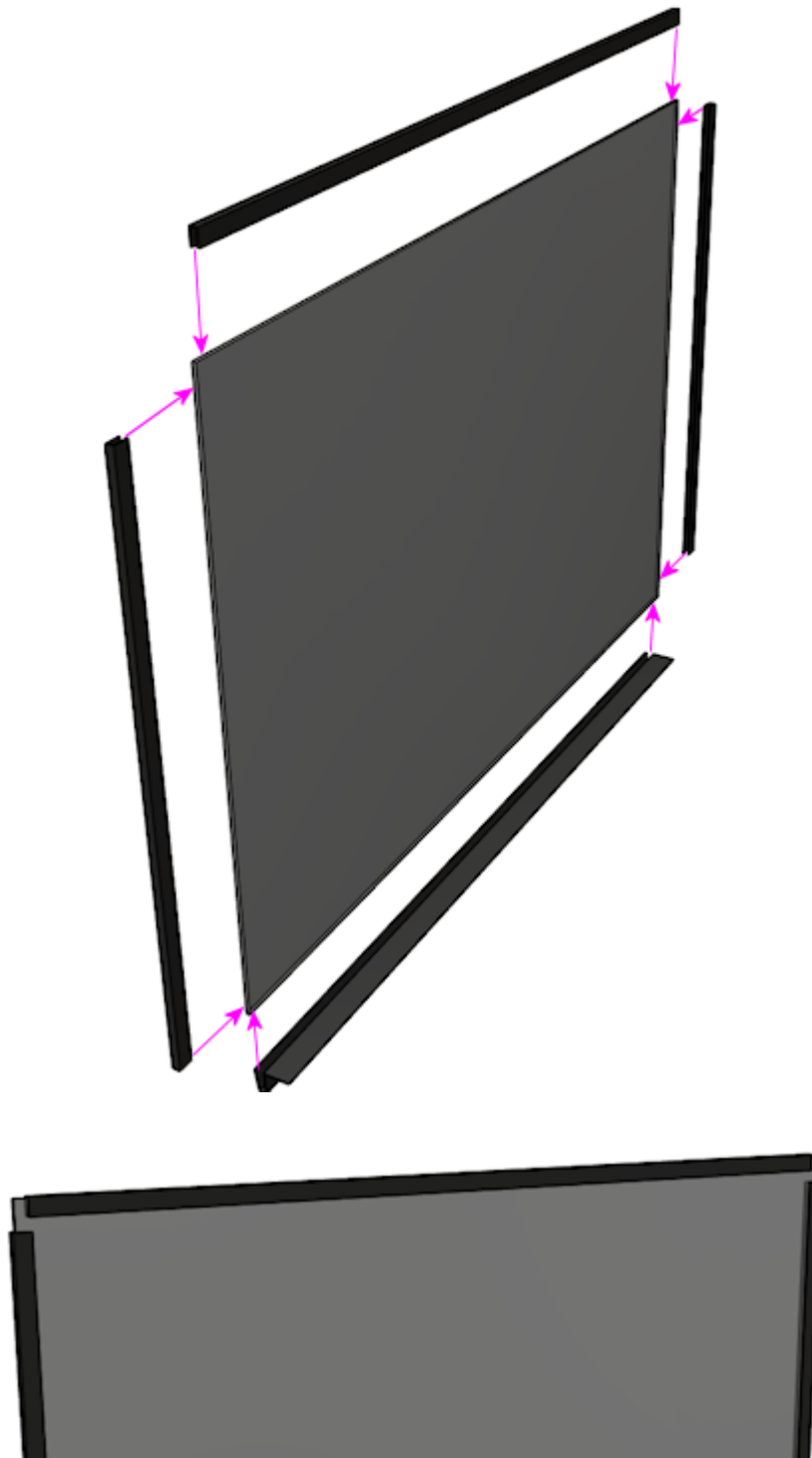
art-impooverished one on a real machine. Even so, a plain black paint mask would do a better job of hiding the TV cutout when the power's off; it would just like a solid dark sheet. You could easily mistake it for a regular translite with really dark graphics that need some backlighting to come to life.

Assembling the trim

The standard WPC translite setup uses four trim pieces around the edges:

- A "lift channel" at the bottom, part 03-8228-1
- A top trim piece, part 03-8228-2
- Two side trim pieces (one for each side edge), Williams/Bally part 03-8228-2

They're all dead simple to install. Each is a plastic piece with a U-shaped channel that the glass/plastic sheet fits into. Just align each piece at the center of its respective edge and press it onto the glass.





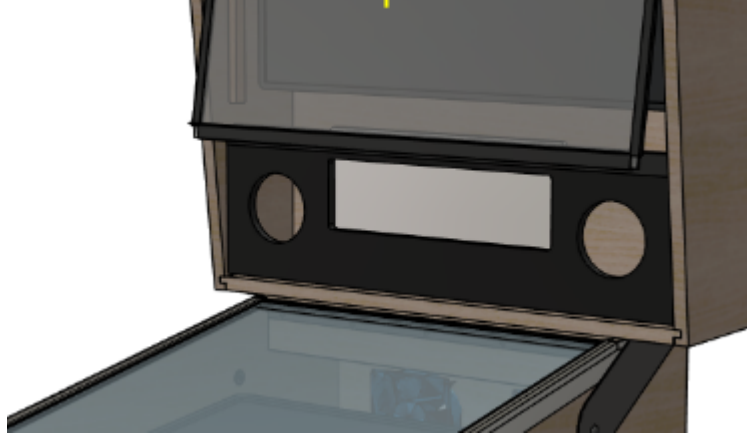
Note that the trim pieces don't cover every millimeter of the edges - there's a little uncovered space at the corners. That's normal.

How to install the translite

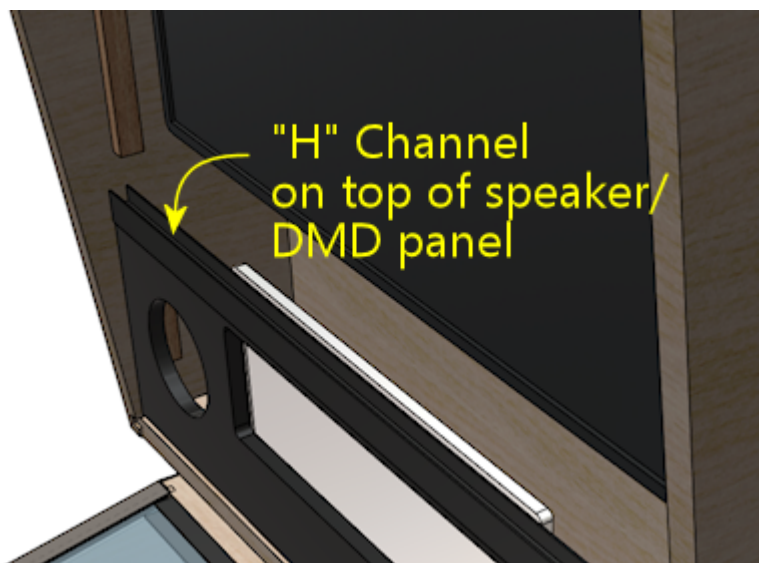
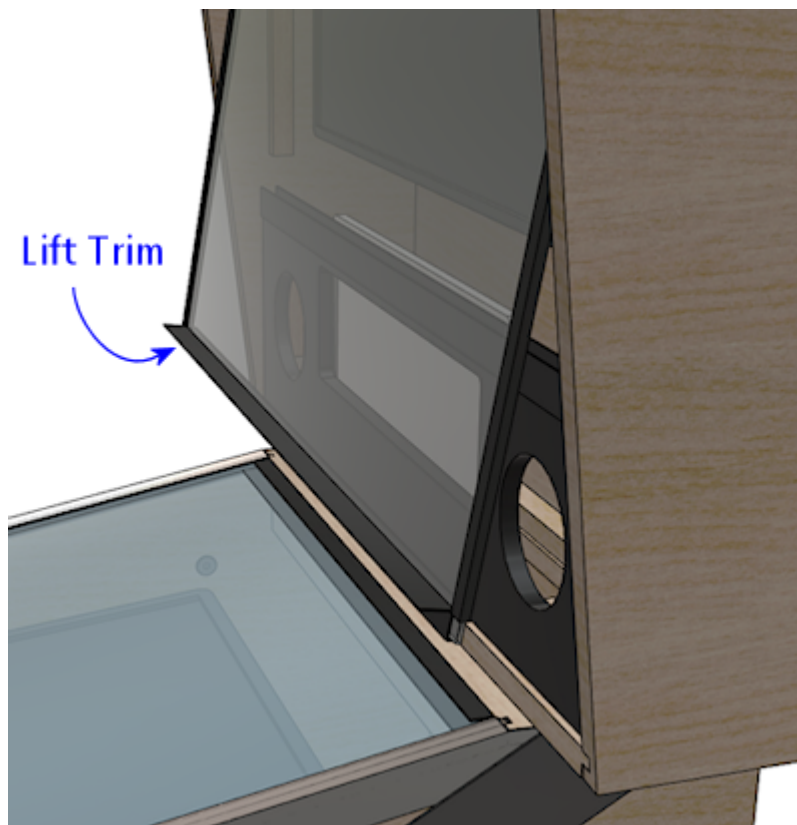
If you installed a translite lock, make sure it's unlocked, with the tab turned "sideways" so that it's not sticking out into the top glass channel. You'll have to use the key to do this. The whole purpose of the lock is that the tab blocks the channel so that the glass can't be removed, but this equally well prevents inserting the glass when the tab is in the "locked" position.

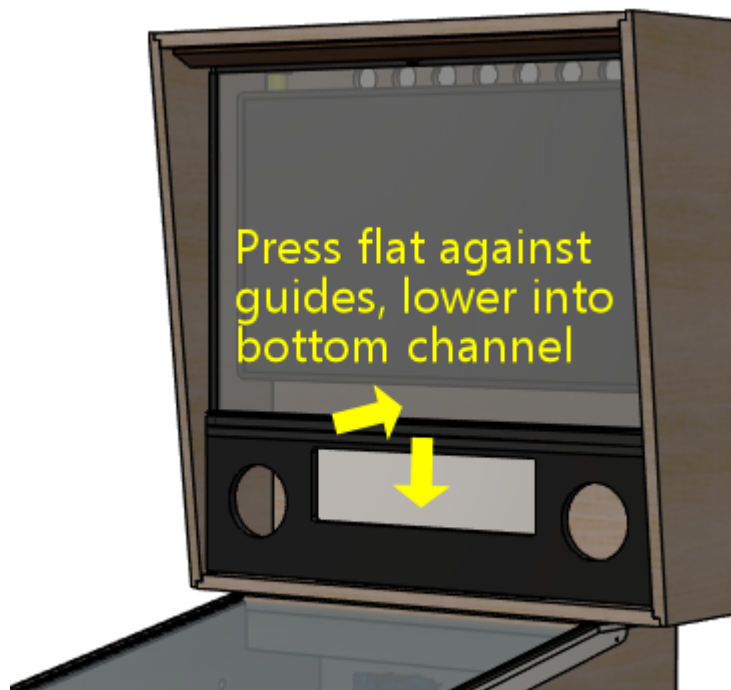
Holding the translite at an angle, lean the top edge against the guides at either side of the backbox, and slide it upwards into the slot at the top.



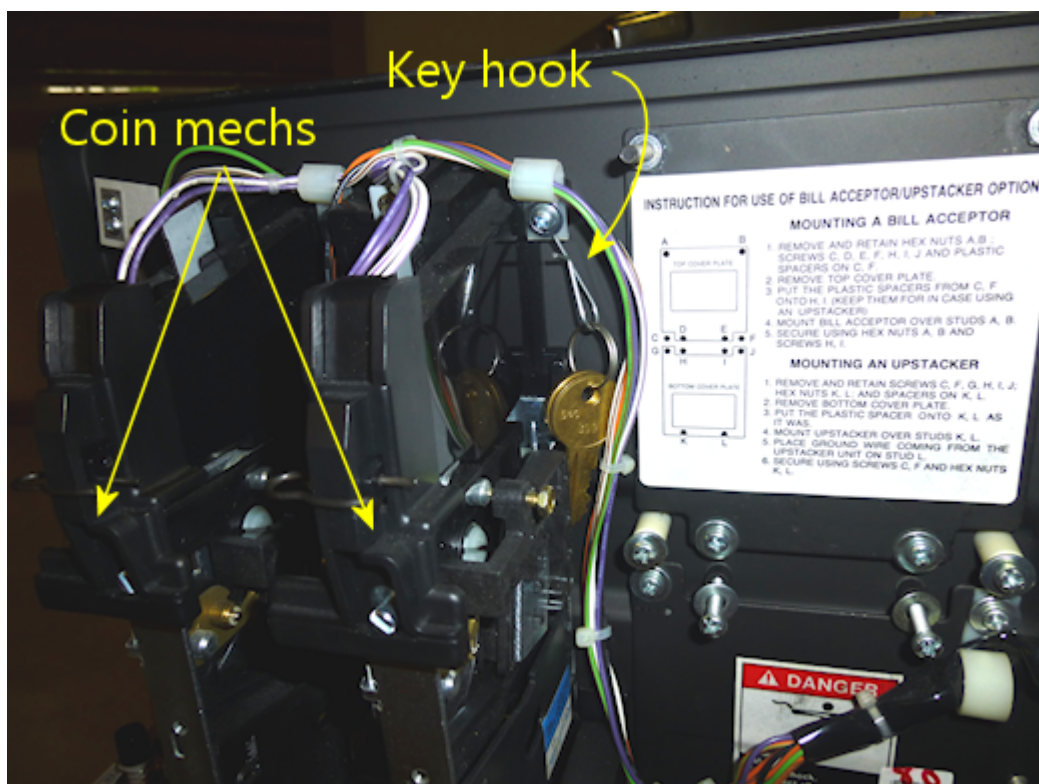


Lift it high enough into the slot that the bottom edge clears the bottom trim channel. Holding it by the "lift trim" at the bottom, move the bottom edge forwards until the translite is flat against the guides, then lower it into the bottom trim channel until it's seated.





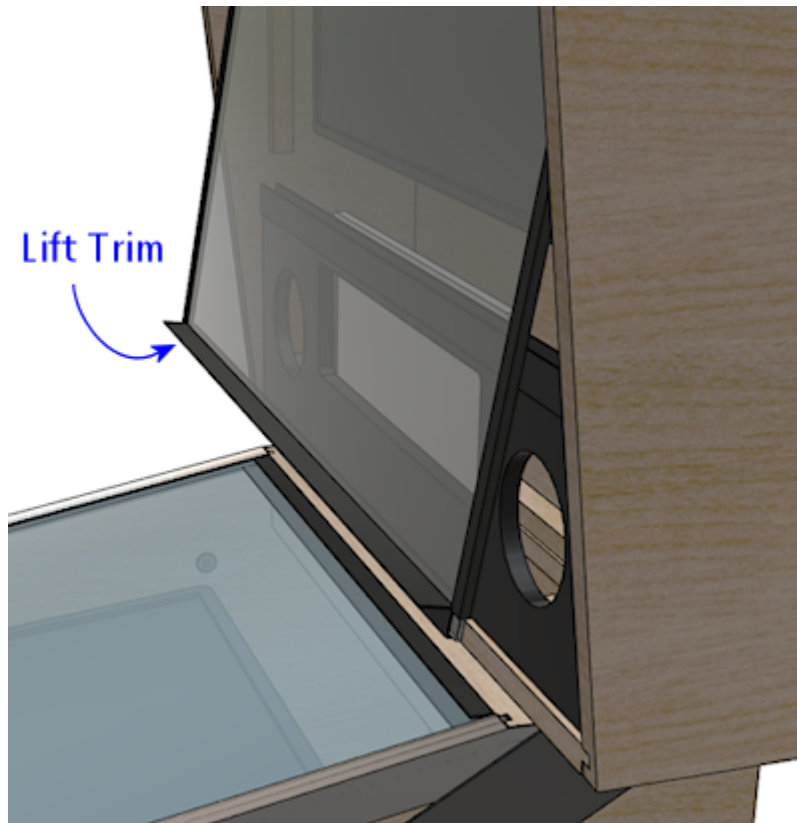
If you have a translite lock, turn the key to the locked position to secure the translite. Most pinball owners store the key inside the coin door, hanging it on a little wire hook that's usually located alongside one of the coin slots.

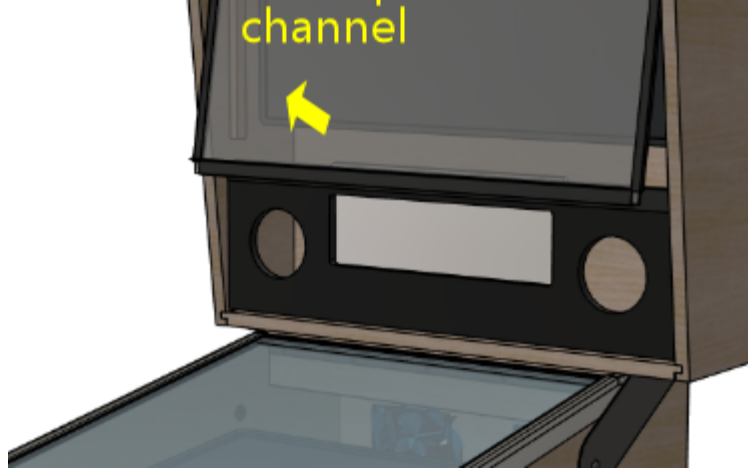


How to remove the translite

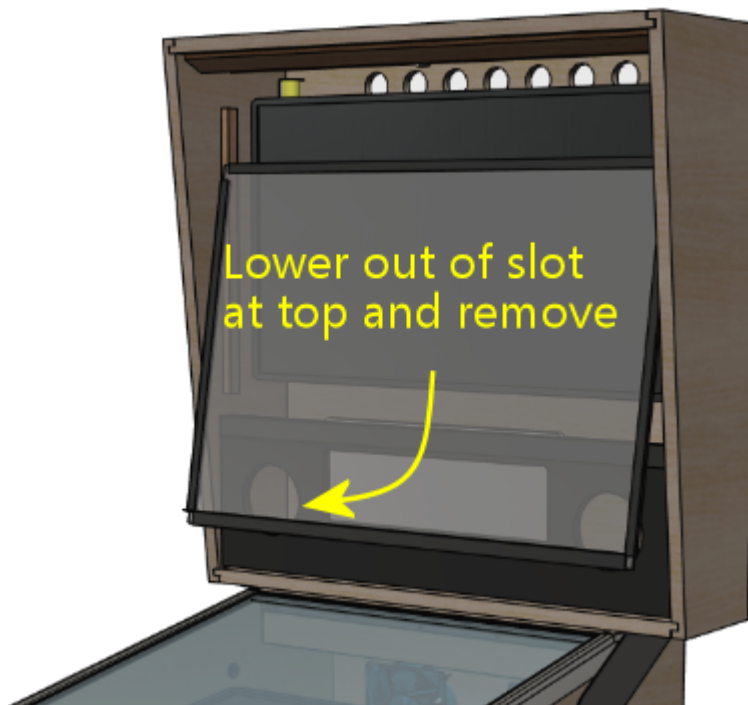
If you installed a translite lock, insert the key and turn it to the unlocked position.

Holding the translite by the "lift trim" at the bottom, slide it upwards until the bottom clears the lip of the bottom trim channel. That lets you tilt the bottom outwards.





Now just lower the translite out of the top slot. It's now entirely free of the backbox trim, so you can remove it.



25. Cabinet Grounding

All of the externally exposed metal parts in your pin cab should be "grounded", meaning that the metal parts are all electrically connected to the Earth ground wire in your AC power plug. This includes the leg bolts, side rails, front lockbar, coin door, and plunger housing.

Grounding serves several purposes:

- **Safety.** If a wire inside the cab carrying voltage ever comes loose, it could come into contact with one of the exposed metal parts, which would create an electric shock risk to anyone using the cab. Grounding all of the metal parts reduces shock risk by shunting any stray voltages directly to ground. It also reduces the chances of a fire or other damage from an electrical mishap, in that a short to ground should quickly trip a fuse or circuit breaker and cut off power at the source, hopefully before anything can get red-hot inside the cab.

This alone makes grounding a must, since the combination of electricity and exposed metal surfaces creates a real risk to human safety in the absence of proper grounding.

- **Static electricity protection.** Semiconductors are extremely sensitive to static electricity. You can destroy a computer chip or transistor just by touching it if you have a static charge on your body. The cab itself can accumulate a static charge as well, which can likewise be a threat to the chips inside the cab. Grounding the metal parts in the cab neutralizes any charge on them, and neutralizes the charge on your body when you touch them.

Grounding the metal parts is a huge convenience any time you're working inside the cab, because it lets you discharge any static on your body simply by touching the side rails or lockbar. That greatly reduces the chance that you'll zap any chips you touch.

- **Radio interference shielding.** Virtual cabs have computer motherboards and other electronics inside that both generate radio frequencies and can be affected by radio waves in the air. The big metal parts in a cab can act like antennas to transmit and receive this energy. Grounding the metal parts prevents that, and reduces the ability of radio waves to penetrate in or out of the cabinet. This will reduce the chances that your cab will interfere with nearby Wi-Fi networks or garage door openers or cell phones, and it'll help avoid things like picking up local radio stations on your cab speakers.

When to install

Grounding is infrastructure work that should be done early in the build. The best time is after you've assembled the cabinet body and installed the metal trim parts, and before you've started installing the internal components (PC, TV, feedback devices). Running the grounding wire is easier while the cabinet is still mostly empty.

Parts

The ideal type of wire to use for grounding is *flat copper braid*. This is a bare (no insulation) wire bundle with a large number of small wires woven into a wide, flat braid.





Braided wire is great for this because it has high current capacity, and the flat form factor makes it easier to connect to metal parts by providing a large surface for contact. The high current capacity is important because the whole point is to carry the large surge current that would occur if a power supply voltage is ever shorted to ground. The ground wire has to be robust enough that it won't melt before the fuse or circuit breaker trips and cuts off the power at the source.

I'd recommend 1/4" (or larger) tinned copper braid. 20 or 25 feet should be adequate. You can find this at electronics vendors, Amazon, or eBay.

What is Earth ground?

Earth ground is pretty much what it sounds like: an electrical connection into the soil. If your house is wired properly, there's a big metal stake driven into the ground somewhere in or around your foundation, and the stake is wired to the "third prong" in all of the AC outlets in your house. In the US, that's the round prong at the bottom of the plug. (In older houses, a buried water pipe might serve in place of a metal stake, and in older houses still, where the old two-prong outlets are installed, you might not have any Earth grounding at all.)



For the purposes of a pin cab, we rely on the house wiring to provide the connection to Earth ground, via that third prong on the AC plug.

How to connect to the Earth ground

One end of your grounding wire needs to connect to the Earth ground in your house wiring. To get there, we can piggy-back on the AC power connection you're using for your PC power supply. That should have a three-conductor power cord, since your PC power supply needs to be grounded.

There are several ways to tap into that ground connection. The best option will depend on your setup and the type of power supply you're using, so look these over and pick the one that works best for you.

Option 1: ATX case

If your PC power supply has a bare metal case, the case itself can serve as the ground connection.

Any ATX power supply with a metal case will have its case connected to Earth ground, for the same safety reasons we need to ground the metal parts on the pin cab. If the case isn't painted, you should be able to get good electrical contact to ground using with the case itself.

It's a good idea to test this with a multimeter, to make sure there's not some kind of coating that insulates the case. With the PSU unplugged from power, and your meter set to resistance (Ohms) mode, measure the resistance between the ground prong on the power supply's AC wall plug (in the US, that's the round bottom prong) and random points on the exterior of the case. It should read close to zero Ohms (you might see a very small resistance on the order of 1 ohm or less). If you get consistent near-zero readings at various random points on the case, the case will make an excellent grounding connection.

To connect the ground braid to the power supply case, you can simply pin the braid under the power supply, pinching the braid between the PSU and the cabinet surface where you're installing it. As long as the PSU is attached tightly to some surface, pinning the braid under it should keep the braid in place and provide a nice solid electrical connection.

Option 2: ATX case screw

If your PC power supply has a painted or lacquered case that prevents the first approach, you might still be able to use the case, by connecting to one of the mounting screw holes - the screw holes intended to be used for mounting the PSU inside a PC tower or desktop case. Even if the case is painted, there's a good chance that the threaded screw holes will be conductive.

You should be able to check this visually. If the threads look metallic, this is probably a good place to connect.

If the visual check looks promising, try the multimeter test. With the PSU unplugged from power, and your meter set to measure resistance (Ohms), test the reading between the ground prong on the PSU's AC wall plug (the bottom round prong, in the US version) and the threads in the screw hole. If you see a reading close to zero Ohms, you have a good place to connect ground.

To use this method of connection, you'll need:

- About 2 feet of 14 gauge (or thicker) wire
- An unpainted metallic #6-32 x 1/4" machine screw
- A "ring" terminal that fits around the screw and that fits the wire

To install:

- Cut the wire to about 2 feet
- Attach a ring terminal to each end (by crimping or soldering, according to what kind of ring terminal you're using)
- Slip one of the ring terminals over the screw, and screw it tightly into the case
- Slip the other ring terminal over a wood screw, and then screw it *through the grounding braid* and into the cabinet wall

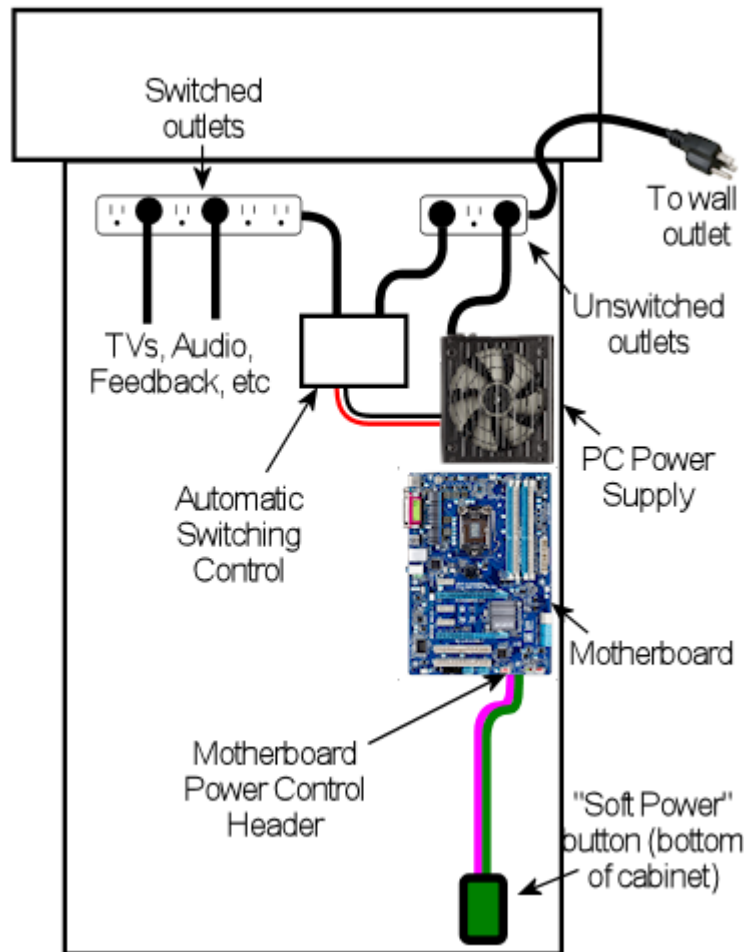
The last step can wait until you're ready to install the power supply in the cab. The dangling ground wire should serve as enough of a reminder, but put it on your to-do list if you think you might forget! And when you're running the grounding braid around the cabinet, make sure you bring it close enough to the area reserved for the power supply that the ground wire will be able to reach it easily.

Option 3: AC plug

If you can't find a way to make a connection to your ATX power supply, you can connect directly to the power line. I don't recommend this approach unless you have

some electronics experience, since it requires cutting into the main AC power wiring. This approach also has the downside that it doesn't make the ground connection as permanent as the others, since it uses a removable plug. Someone down the road might decide to unplug it because they want to use the outlet for something else, without realizing how important it is to leave it in place.

In order for this approach to work, you'll need a setup that follows the basic plan we outlined in Chapter 11, Power Switching. Something like this:



Specifically, you'll need an unswitched power strip that connects directly to the wall outlet. All of the outlets on that power strip will have a connection to Earth ground through the wall outlet plug, so we can get the Earth ground connection we need inside the cab via an unused outlet on the power strip.

The idea here is simple: we need a power cord that *only* has a connection to the ground prong in the outlet. I can suggest three ways to achieve this:

- Buy a "ground plug", such as a "Desco universal ground connection" or "StaticTek banana jack outlet plug ground adapter" (try Amazon or eBay). These are AC plugs with dummy prongs for the two power prongs, and a ground prong that connects to a banana jack or similar connector. They're designed to be used with anti-static wrist straps and mats for doing electronics work. You'll also need a banana plug that fits the jack. Attach a wire (14 gauge or thicker) to the banana plug; plug it in the jack and secure it; and connect the other end of the wire to the braid.

This approach has the advantage that you can't get the wiring wrong, since the ground plug only has a connection to the one ground prong. The downside is that the banana plug isn't permanently installed, so it could fall out, disconnecting all of the ground connections you went to all this trouble to

install. If you go this route, I'd find some way to permanently secure the plug so it can't fall out, perhaps with electrician's tape or heat-shrink tubing.

- Buy a replacement power supply cord (making sure it's the 3-prong type). This has a regular AC outlet plug at one end and three insulated wires (black, white, and green) coming out the other end. The **green** wire is the one that connects to the Earth ground prong. Use wire nuts to cover the white and black wires, which you **don't** want to connect to anything, and secure with electrician's tape. Connect the green wire to the braid.
- Buy a replacement power plug (e.g., Leviton 3W102-E, GE 54301 household plug). This is similar to the above but doesn't have any wires attached - it's *just* the plug, with screw terminals to attach wires. I like this option a little better than using a cord because you don't have to secure any stray wires. Simply connect a 14 gauge (or thicker) wire to the ground screw terminal (which is usually indicated by a green screw, or might simply be labeled "ground" or "Earth"). Leave the other two terminals unconnected. Connect the other end of the wire to the braid.

For all of these options, plug the plug into a free outlet on the unswitched power strip. Connect the ground wire from the plug to your braid with a "ring" terminal: connect the ring terminal to the wire (by crimping or soldering, for example), slip the ring over a wood screw, and drive the screw *through the braid* into the cabinet wall or floor.

Whichever type of ground plug you choose, it would be a good idea to do something to lock the plug into the outlet it's using, so that it doesn't fall out on its own and so that you don't remove it while working on something and forget to put it back. This is the crucial link for all of the grounded metal, so it should always be connected. Wrap a couple of loops of electrician's tape around the plug and the power strip, for example. At the very least, put a big "do not unplug" placard on it.

Option 4: Tap into the power strip

If you're confident that you know what you're doing, there's a better alternative to the approach above: tap directly into the power strip's internal wiring. It's better in that it's not easily undone (unlike the plug-in approach above, where someone could unplug the plug, thinking it's not important). But it's dangerous unless you know exactly what you're doing, since it requires modifying the power strip.

The idea is to connect an additional wire directly to the ground wire in your main unswitched power strip.

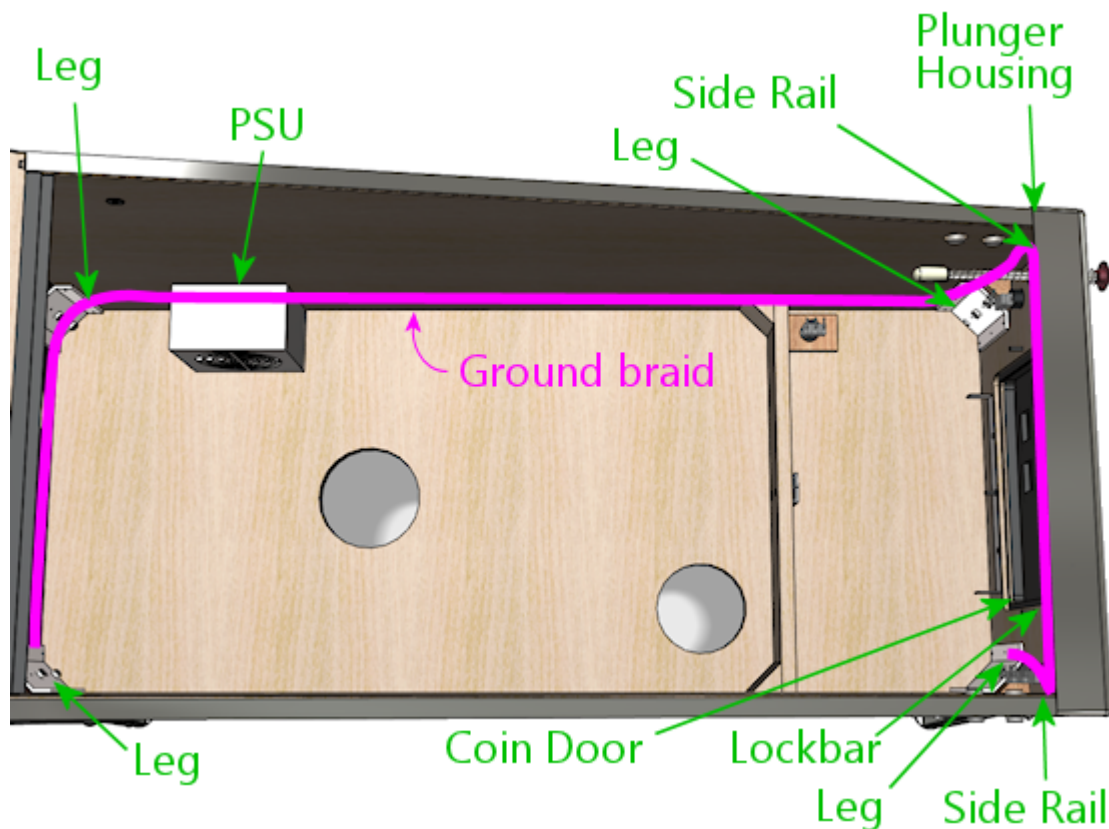
- Open up the power strip (by removing its case)
- Identify where the ground wire from the cord connects to the internal wiring
- Connect a length of 14 gauge (or thicker) wire to this point (using whatever technique is appropriate to the way the power strip is constructed: solder the new wire to the existing wire, add it to the existing screw terminal, or whatever else works)
- Find a way to route the new wire out of the power strip's case, perhaps by drilling a hole somewhere for it
- Reassemble the case with the newly added wire routed through to the outside
- Connect a ring terminal to the other end of the wire
- Slip a wood screw through the ring terminal, and drive the screw through the braid into the cabinet wall

How to connect cab parts to the ground braid

The basic technique is to run a single, uninterrupted braid around the perimeter of the cab, bring it into contact with each metal part that needs to be connected.

The reason it's best to use a single run of wire is that it greatly reduces the chance of severing the connection to multiple parts. Consider what might happen if you daisy chained *separate* wire segments from one metal part to the next: suppose the Earth ground connects to A, and A connects to B, and B connects to C. If the connection between A and B gets disconnected for some reason, you lose not only the connection to B, but also the connection to C. With a single braid, in contrast, the only way that could happen is if the braid itself were to break, which is highly unlikely.

Here's a suggested routing:



Use staples to fasten the braid to the cabinet wall every few inches between connections, so that it doesn't flop around.

Remember that the ground braid is uninsulated, so you don't want to let it come into contact with exposed terminals on any powered devices. Ideally, you should avoid having any bare wire or exposed terminals (other than the ground braid) in the first place, since they're inherently dangerous. If possible, cover any exposed terminals that are present on devices you install with some kind of insulator, such as heat-shrink tubing, electrical tape, or a plastic cover.

To connect an individual metal item to the braid, all you have to do is bring the braid and the metal into contact.

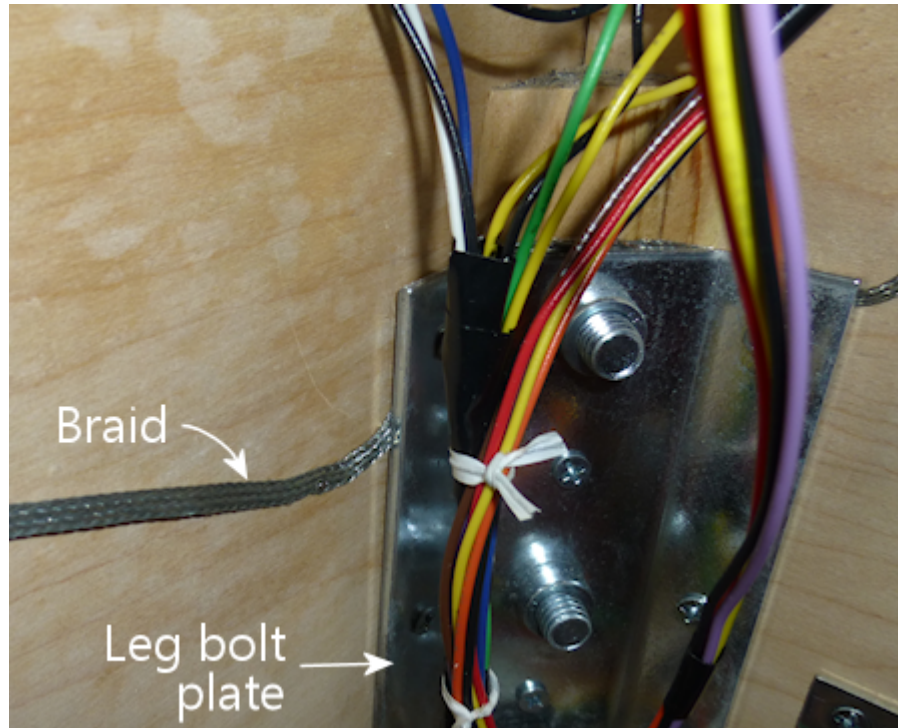
- For anything that has a large surface that fastens tightly to the cabinet, a great way to accomplish this is to run the braid under that part, sandwiching the braid between the part and the cab. This provides a large contact area, ensuring a good electrical connection, and secures the braid in place

mechanically. It also has the virtue of being easy to set up.

- Alternatively, if there's a place where a metallic screw is attached to the item, you can drive the screw through the braid, or pin the braid under a washer held down by the screw.

Legs

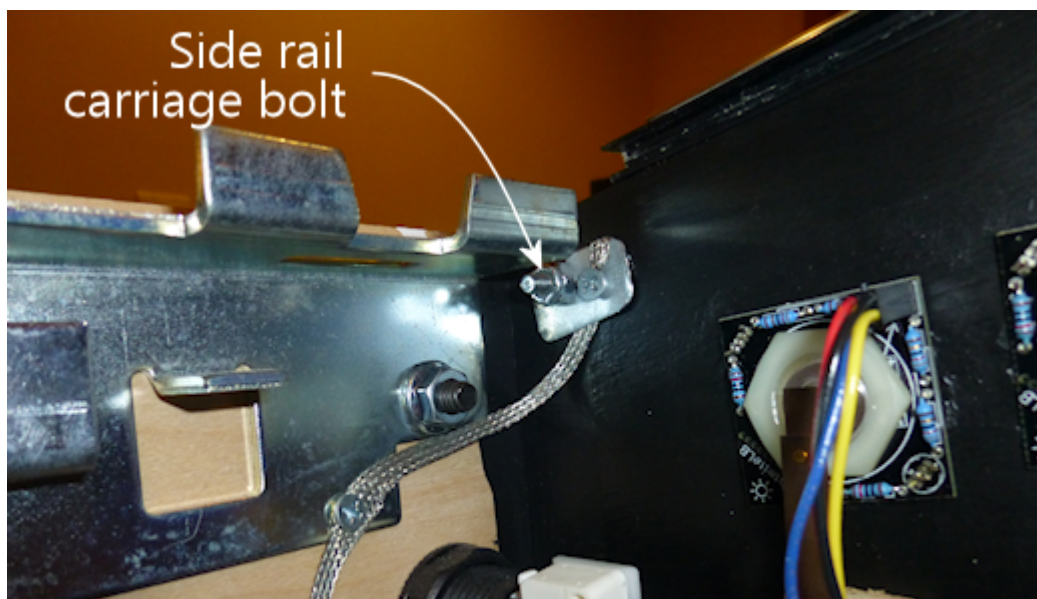
Simply run the ground braid under each leg bolt plate.



Side rails

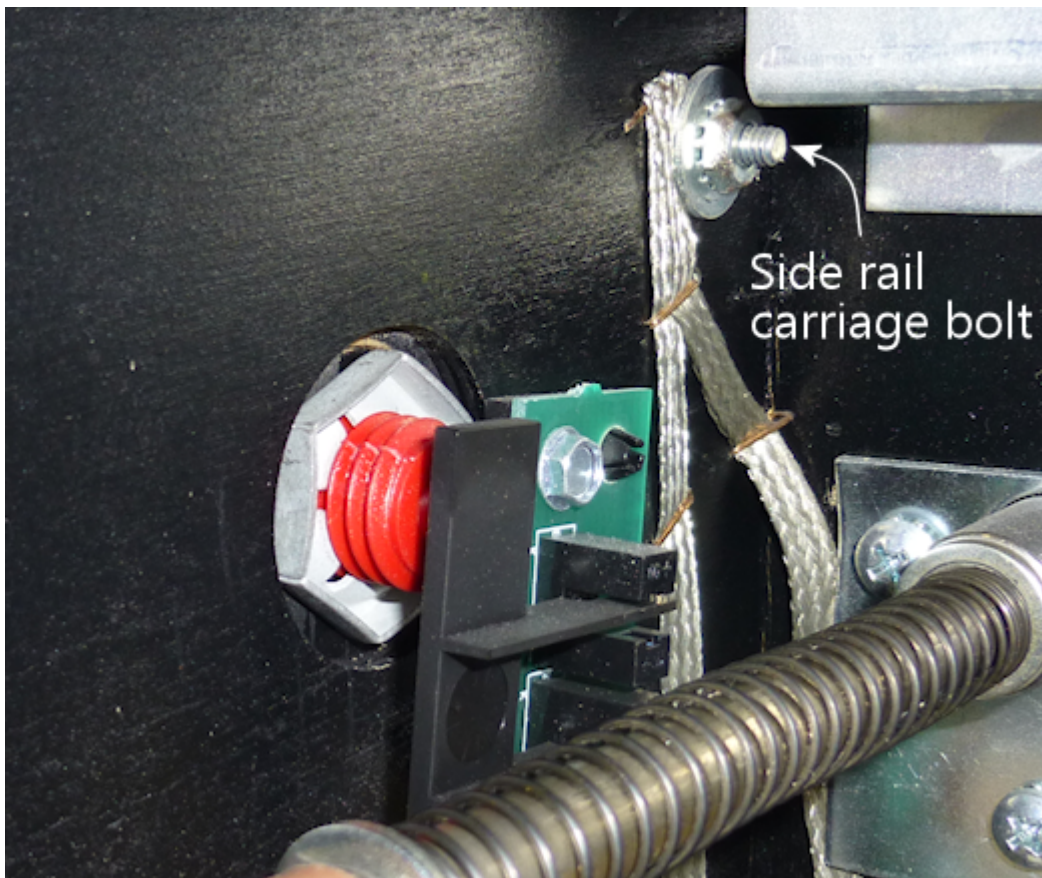
The side rails are held on by carriage bolts at the front. Those are metallic, and they're in contact with the rails, so we can ground the rails by grounding the bolts. The bolts don't by themselves offer much surface area to make contact with the ground braid, though, so we have to add something to serve as a connector.

My approach was to use a small metal plate with two holes, one for the bolt itself, and a second for a wood screw. I ran the braid under the plate, and fastened the wood screw through the braid to ensure a solid electrical connection.



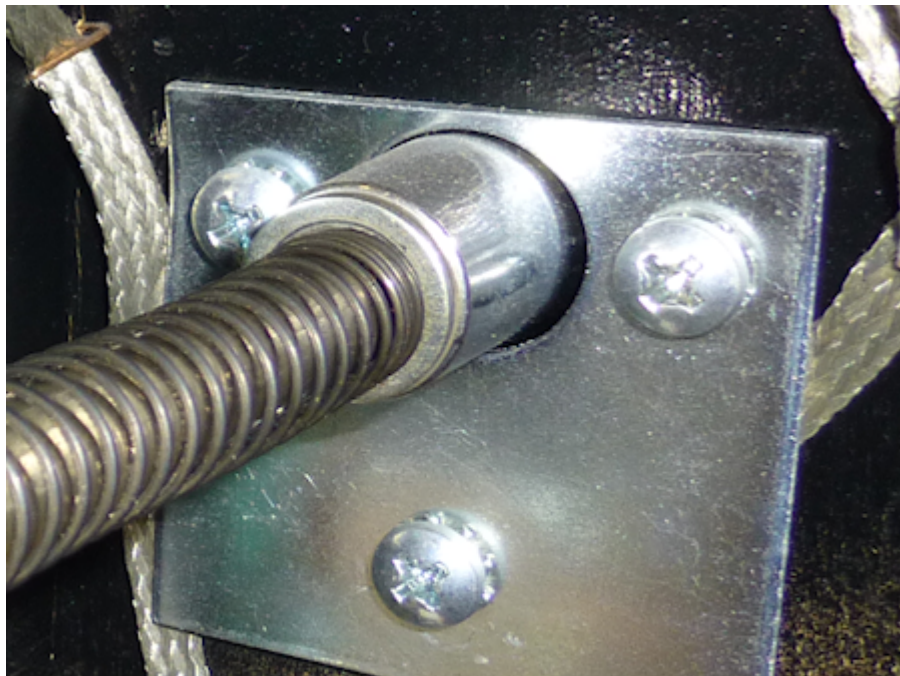


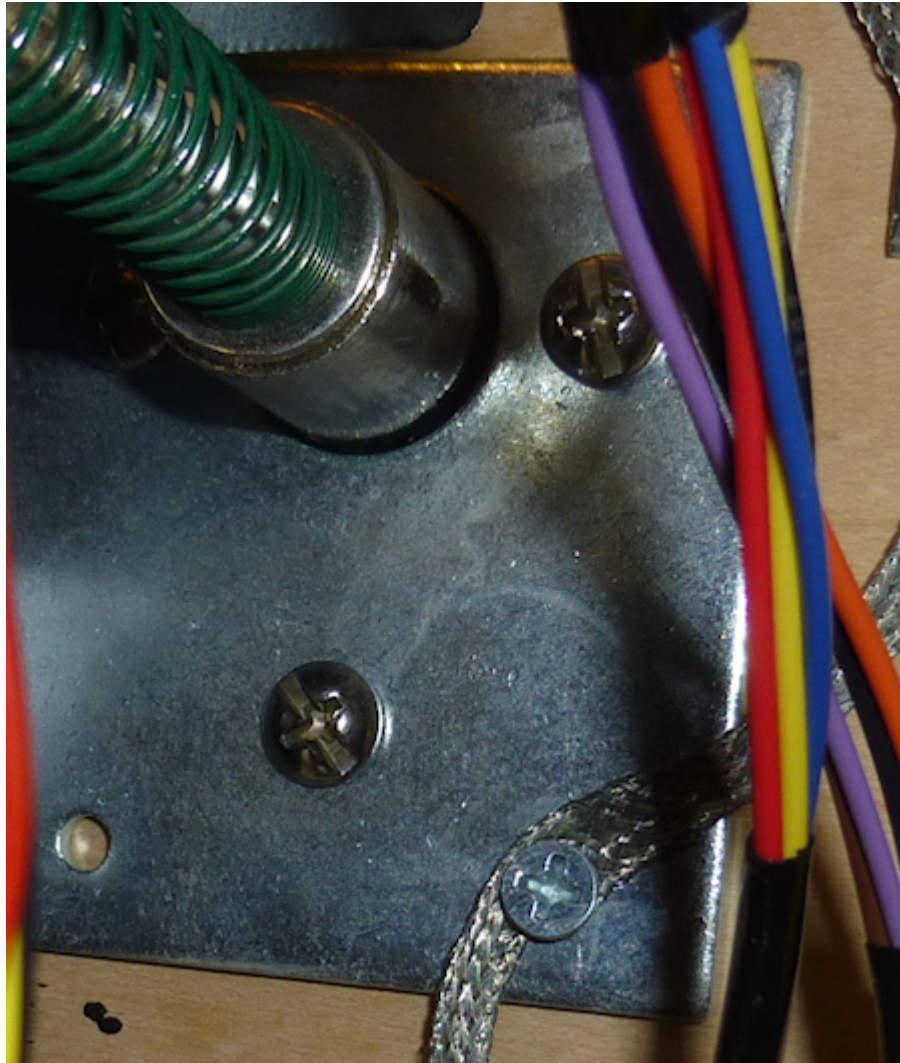
On some of the real machines, they simply pin the braid under a washer.



Plunger housing

Run the ground braid under the mounting plate, or fasten it with a wood screw through one of the free holes in the plate.





Lockbar

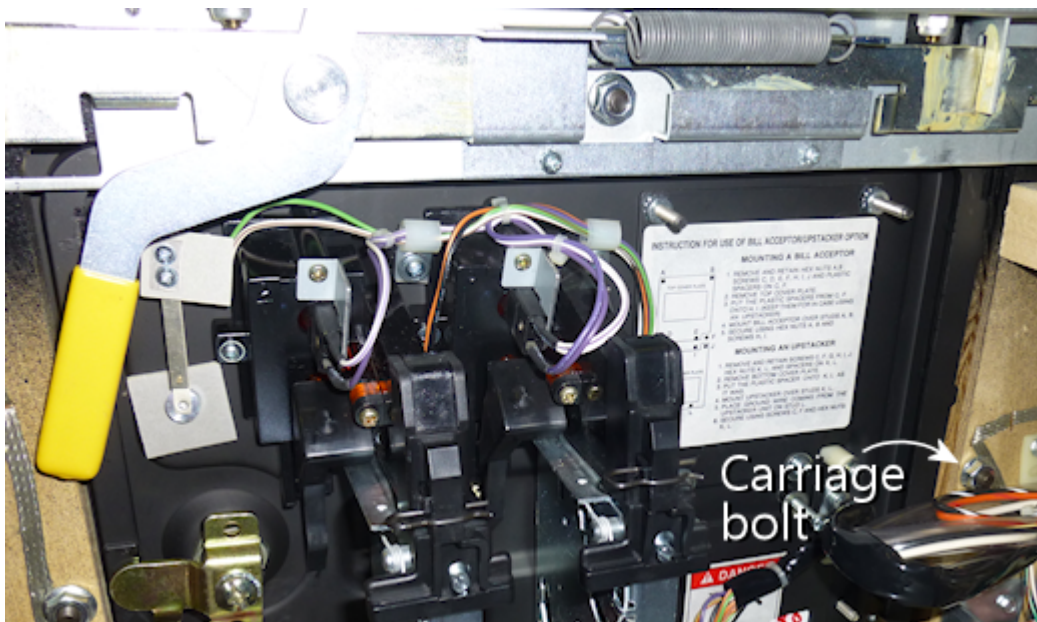
Route the braid under a portion of the lockbar receiver where it attaches to the front wall, or fasten it with a wood screw through one of the free holes in the receiver.

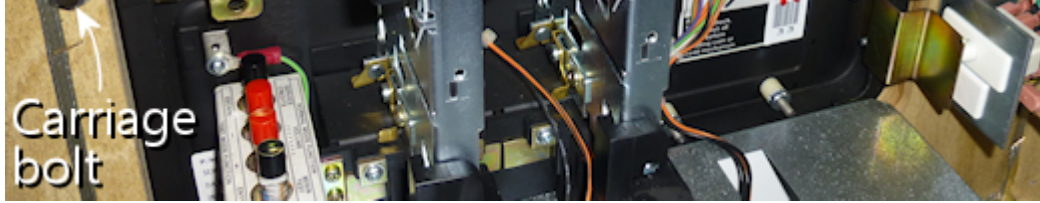




Coin door

You can ground the coin door through the carriage bolts that attach it. (It'll also be grounded indirectly through the lockbar receiver, assuming you've grounded that, since the top coin door bolt also is in contact with the receiver.) Run the ground braid alongside one or two of the carriage bolts on either side of the door, and pin it under a washer.





Backbox

There's not any metal trim in the standard backbox setup, so you might not need to extend the ground wire there. However, the real machines do, because they have some hidden metal pieces that benefit from grounding. In particular, they place a metal grating over the vent holes along the top of the back side of the backbox, primarily to serve as radio frequency shielding. That needs to be grounded to be effective as shielding. They also run the ground braid under the metal backing plates that mate with the carriage bolts that fasten the hinge arms, as safety grounding for the exposed carriage bolt heads. (I guess there actually *is* some metal trim on the backbox, if you count those bolts.)

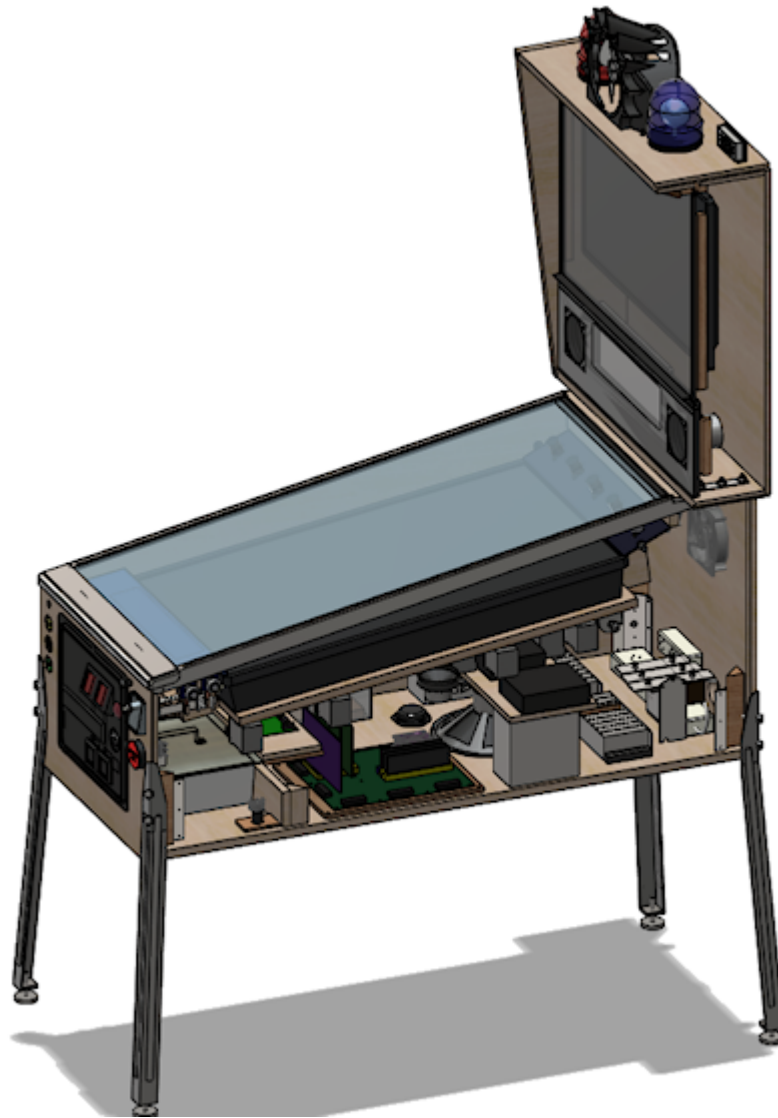
If you do want to run a ground wire to the backbox, I'd use a separate braid loop in the backbox, and connect it to the braid in the main cabinet via a run of regular hookup wire (14 gauge or thicker). The reason to use hookup wire to bridge the sections is that this portion will need to be long enough to cover the added distance when the backbox is folded down.

Testing

Before declaring the grounding project complete, test that you have a good connection between the metal parts and the ground plug on your main power inlet.

Set your multimeter to resistance (Ohms). With the power unplugged from the cab, measure the resistance between the ground prong on your main AC power plug for the cab and each of the exposed metal parts. It should read close to zero Ohms in each case.

26. Inside the Cabinet



The modern virtual pin cab can be pretty complex on the inside. A decked-out cab can actually have more equipment packed into it than a real pinball machine, which is kind of perverse given that this is all about software simulation. But it makes sense when you consider that we're not just building a computer; we're building a computer/pinball hybrid. The computer part by itself has more to it than most desktop systems, because of the extra monitors and the specialized input/output peripherals. And the pinball part includes a pretty large subset of a real machine.

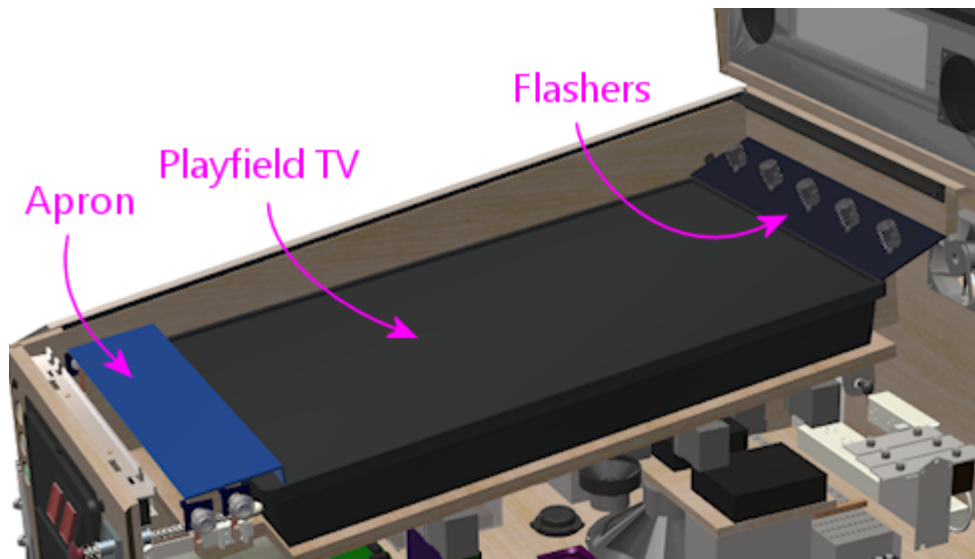
With so much to install, making everything fit can be a challenging 3D puzzle. I'd like to be able to present a simple, one-size-fits-all layout here, but that's not really possible. Pin cabs are too individualized. But I can at least offer one possible solution. In this section, we'll walk through a model pin cab that includes just about everything I can think of, and look at where each major element goes in this setup. The model takes into account the space constraints, and it also follows my philosophy of serviceability, meaning that it's designed so that everything can be accessed fairly easily for repairs and upgrades, even after the machine is fully built.

The arrangement described in this section is based on my own cab, so I consider many of the design decisions to be tried and true. It's not an exact replica, though. I've made some revisions in an attempt to improve things that weren't ideal in my original design. I also made room for additional equipment that's not in my build. My cab is pretty decked out, but for the purposes of this section, I've tried to imagine an "ultimate" cab with all of the toys.

This is, of course, not the only possible solution to the question of how to arrange things, and I'm sure it's far from the best solution. Some of it might not work at all for your setup, given that you might have completely different constraints from your PC packaging or TV size. So I'll try to explain the rationale behind each element's placement, so that even if you can't use the exact layout as presented, you can at least gain some insight from it to use in formulating your own layout.

Playfield, apron, and flashers

These elements (or some subset of them) form the top layer in the main cabinet.



The basic arrangement is pretty straightforward, but the details can be surprisingly hairy. First, there's the placement of the TV: do you place it flush with the top of the cab, or recessed like a real playfield? How far in? At what angle? All the way at the front, or set back to make room for the plunger? These are all among the most frequent questions that new cab builders ask. You can see from the diagram what I prefer, but this is a matter of aesthetics, and there are other schools of thought. Second, once you've decided upon the desired look, you still have to implement it physically. That's trickier than it might look, especially if you want to fulfill my admonition to make the machine serviceable (Chapter 6, Serviceable Design). Serviceability requires that the TV be easily removed. You can probably guess I'm not in favor of just nailing it in there and calling it done.

There's enough to this subject that we've given it its own chapter, Chapter 29, Playfield TV Mounting.

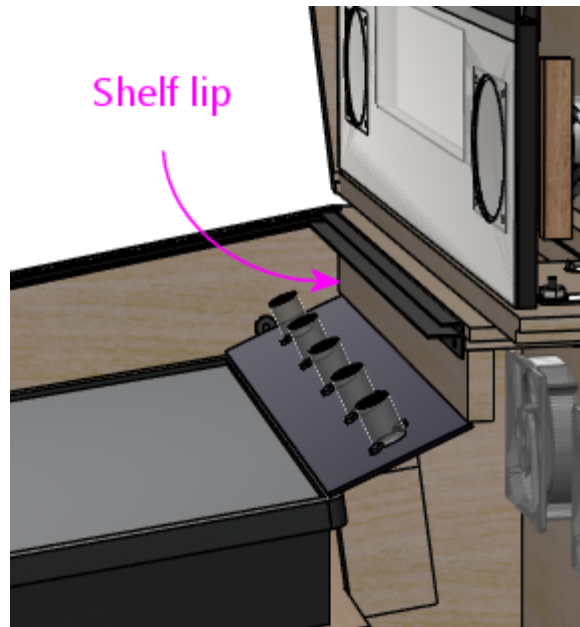
It's extremely important to plan out exactly where the TV goes before you start arranging anything else inside the cabinet. The TV assembly forms a ceiling that constrains the vertical space available to everything else, so you need to know where that is.

If possible, you should actually install the TV early on, not just make plans, so that you can see the space it delineates for real rather than just as measurements. But don't do that unless you're using a mounting that's easily removable, because you won't want the TV in the way while you're installing everything else. If you use a mounting system like the one I outline in Chapter 29, Playfield TV Mounting, you'll be able to install and remove the TV with little effort.

Some notes on the flasher panel. I've depicted the back panel with the traditional five flasher domes. Each is a clear plastic dome with a 3W RGB LED inside. This is

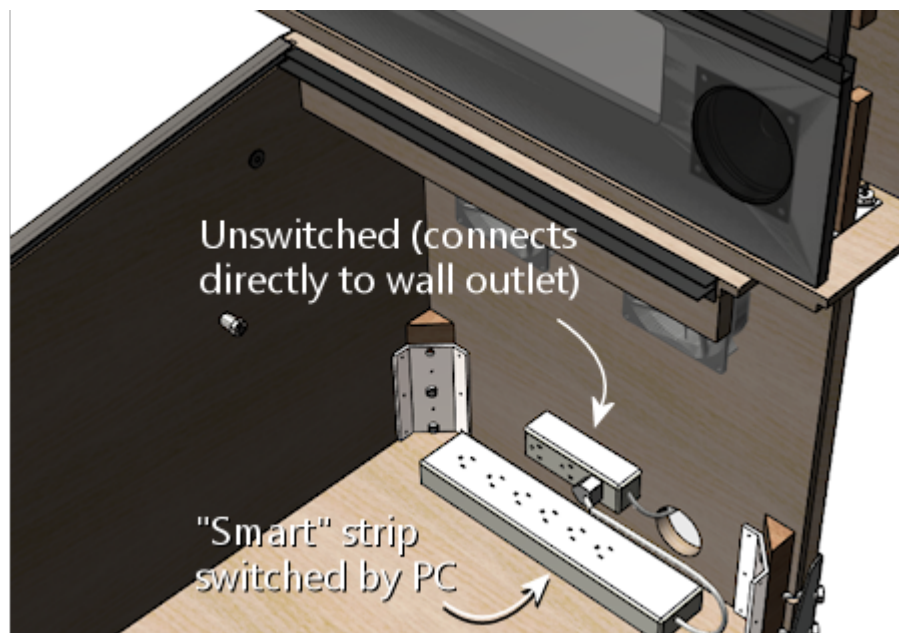
such a ubiquitous setup that the most popular pinball software (Visual Pinball with DOF) is programmed to assume it's there. However, some people replace the five-flasher panel with an array of individually addressable LEDs - sort of a coarse dot-matrix display. The physical setup for that is basically the same; just remove the flasher domes and substitute addressable LED strips or arrays. Some people also do both, by installing a five-flasher panel as shown and adding an addressable LED strip or two, across the top and/or bottom. See Chapter 56, Flashers and Strobes and Chapter 65, Addressable Light Strips.

You could also fit one or more light strips across the "lip" (illustrated below) that sits below the backbox shelf. Keep in mind that there's a standard trim piece for the top glass that also affixes here, so check how that will fit before finalizing plans. See "Rear glass trim" in Chapter 23, Cabinet Hardware Installation.



Power inlet

Okay, let's take the TV out and look inside the cab. We'll start with some simple infrastructure: the power strips. I like to put these at the very back of the cabinet.



Why at the back? For one thing, that's where the power cord customarily comes in. For another, it's a really good fit for the geometry: most power strips are long and narrow, which is a shape that fits nicely along the bottom of the back wall. And finally, it's good to put something low-maintenance back there, because that area is relatively difficult to reach into once everything's assembled. The very back is blocked from above by the backbox shelf, so it's a little bit of an inconvenience to access. It's just reachable enough for plugging and unplugging AC cords, but you wouldn't want to have to get back there with tools if you can avoid it.

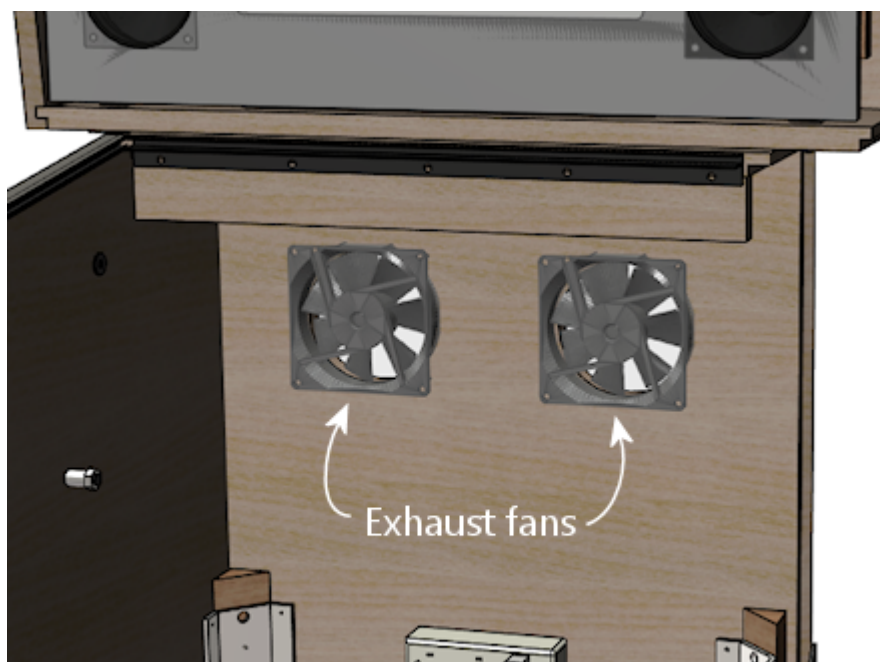
I recommend installing *two* power strips: a small strip that you'll plug directly into the wall outlet, and a second, larger strip that acts as a "smart" strip, providing power to its outlets only when the PC is powered on. All of the accessories (the TVs, audio amplifiers, and feedback devices) plug into the switched outlets, which lets you turn the whole cab on and off with the PC soft power controls. You can implement the switched outlets by buying a smart power strip (they don't design them for pin cabs specifically, but it's the same idea: they're for turning off your monitors and printers when you're not using the computer), or by building your own. This is all covered in much more detail in its own section, Chapter 11, Power Switching.

I'd put the large put strip along the base of the back wall, and mount the smaller strip on the rear wall a little ways above. Put it high enough up that it won't be in the way of the plugs on the main strip. We have enough stuff to pack in here that it's important to think three-dimensionally, so utilize wall space when it makes sense.

The power strips should be secured in place somehow. As always, I'd *avoid anything permanent*, such as gluing them down: think serviceability when choosing installation methods. Adhesive Velcro on the bottom would be a good choice for the large strip mounted on the floor. I *wouldn't* use Velcro for anything mounted on a vertical surface; the glue on it doesn't hold up over time when under constant tension from gravity. I'd use some sort of screw-in brackets instead. Or you could build a little shelf for it (jutting out from the rear wall), and Velcro it to the strip to the shelf.

Rear exhaust fans

As long as we're looking at the back, don't forget the exhaust fans. As mentioned above, the backbox shelf makes this area at the back cumbersome to access when the cab gets fuller, so it's good to get the fans in place early.





Rear wall exclusion zone

After installing the power strips and exhaust fans, there's still a lot of open space on the rear wall. I'd recommend leaving this space unused for now, for several reasons:

- It's hard to reach after the machine is assembled, as mentioned earlier, due to the way the backbox shelf overhangs this area
- There will eventually be a bunch of wires and cables that you'll have to route through this area from the backbox
- If you're using a liftable playfield TV mounting like the one outlined in Chapter 29, Playfield TV Mounting, you'll need to keep most of this space open so that the TV has room to maneuver; plus, the TV overhang will make the space even harder to reach

After everything else is assembled, you can reconsider this space if there's "just one more thing" you want to install and you can't find space for it anywhere else. At that point you'll have a more concrete idea of the constraints on this space, so it'll be easier to decide if it makes sense to mount anything else here. In my cab, I find this space so hard to access that I wouldn't put anything here besides what we've already covered.

Subwoofer

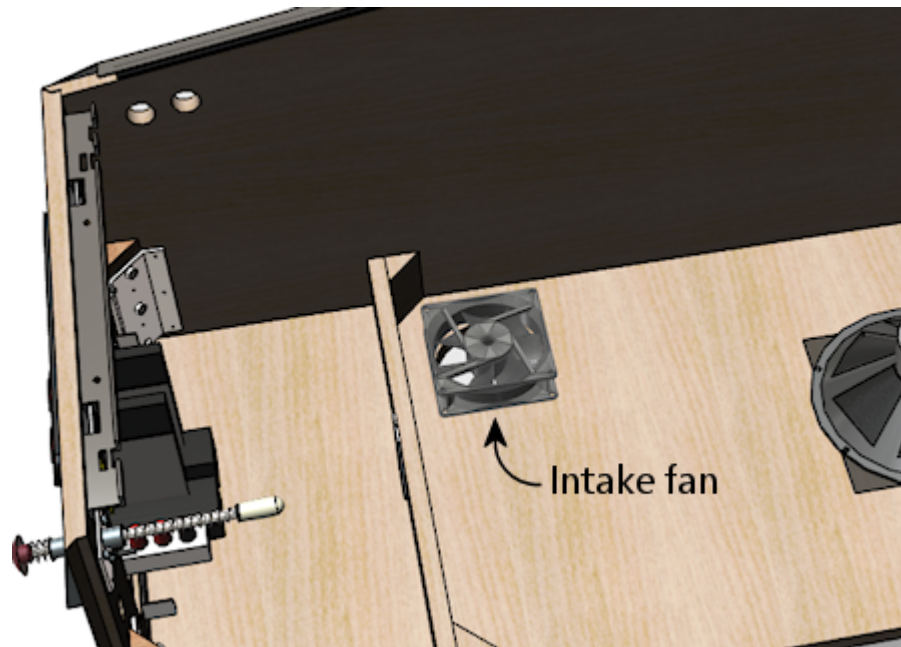
The subwoofer's position is forced by where you placed the floor opening for it. That should be installed next, so that you can take it into account when positioning other things.



Most subwoofers have screw holes around the perimeter of the speaker opening. Use suitable wood screws. If you're using a screen cover, place it between the speaker and the cabinet floor. For a plastic screen, you might want to pre-cut holes where the screws go; driving a wood screw through the plastic can bend or twist the plastic.

Intake fans

As with the subwoofer, the intake fan or fans are constrained to be placed at the openings you made for them, so they should be installed now to ensure that you don't create space conflicts for them later.



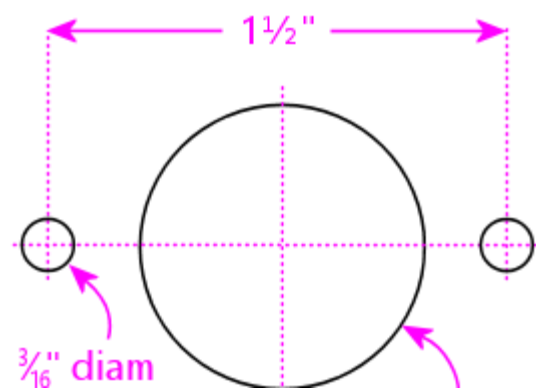
Most PC fans come in square mounting frames (like the one illustrated above) with screw holes at the corners that you can use to secure the fan to the cab floor.

Note that you can buy dust filters for PC fans. Since this is an intake fan, it's a great place to put a filter, to reduce dust buildup inside the cab. Place the filter between the fan and the cab floor.

PC power switch

The SuzoHapp "large rectangular button" (part number D54-0004-5x) is a good form factor for the main power button. It fits in the power switch opening used in the standard WPC plans, and it's large enough that it's easy to operate by feel (which is nice because it's hidden on the bottom of the cabinet, so you want to be able to just reach under and press it without having to see what you're doing).

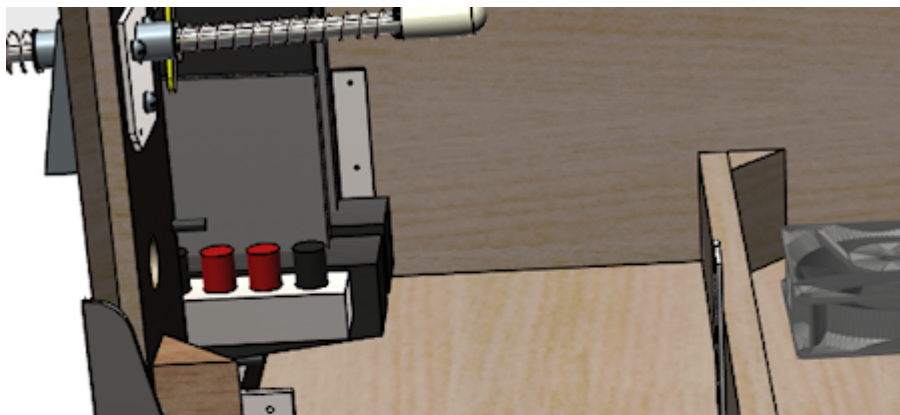
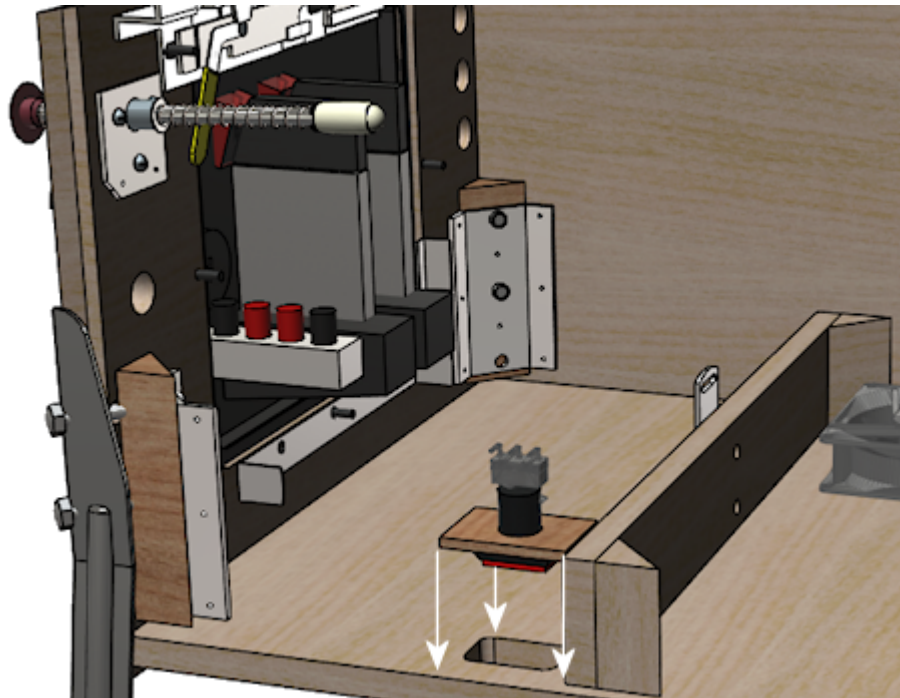
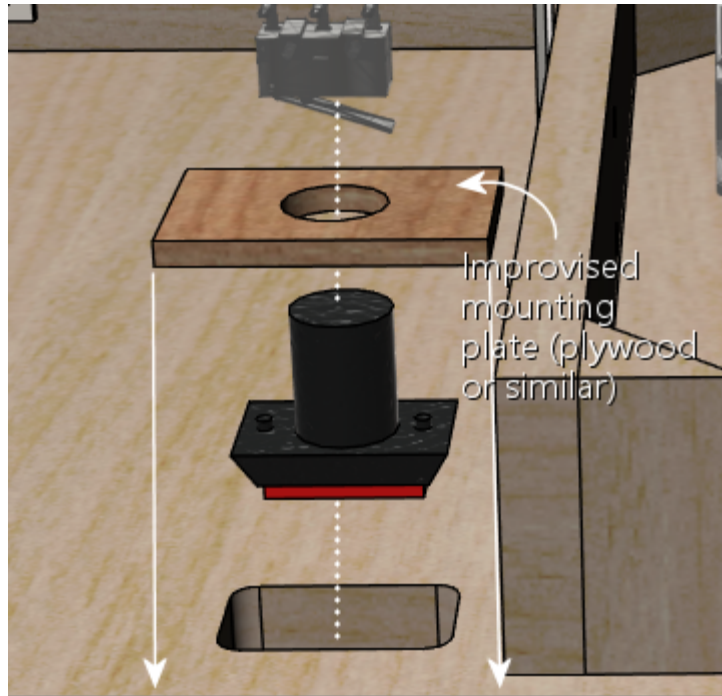
You can install this type of button by creating a small mounting plate using plywood or any other convenient material. Cut holes in the mounting plate using the drilling template below, then assemble as illustrated. Then simply screw the plywood mounting plate into the cab floor from the inside. This will leave the button perfectly recessed in the switch opening.

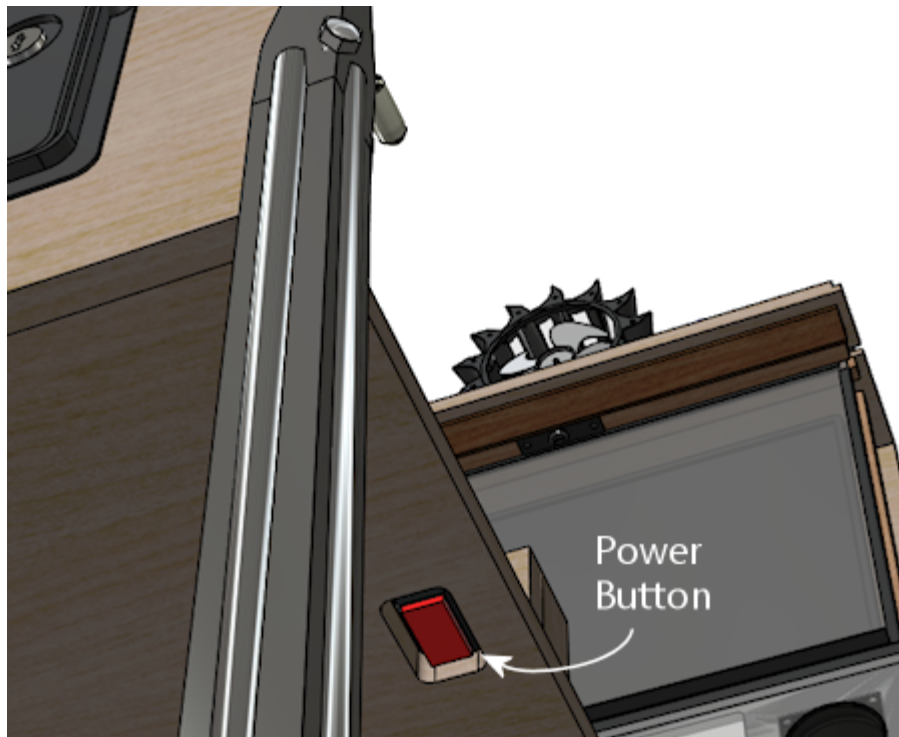


$\frac{3}{8}$ " deep

Drill 1" diam

Drilling template for SuzoHapp large rectangular pushbutton (part D54-0004-5x)





You can easily substitute any of the other similar SuzoHapp pushbuttons (small round pushbutton, square pushbutton) if you prefer. I like the large rectangular button because it fits the opening nicely and it's large enough that it's easy to operate by feel, which is helpful given the hidden location.

Coin door switch

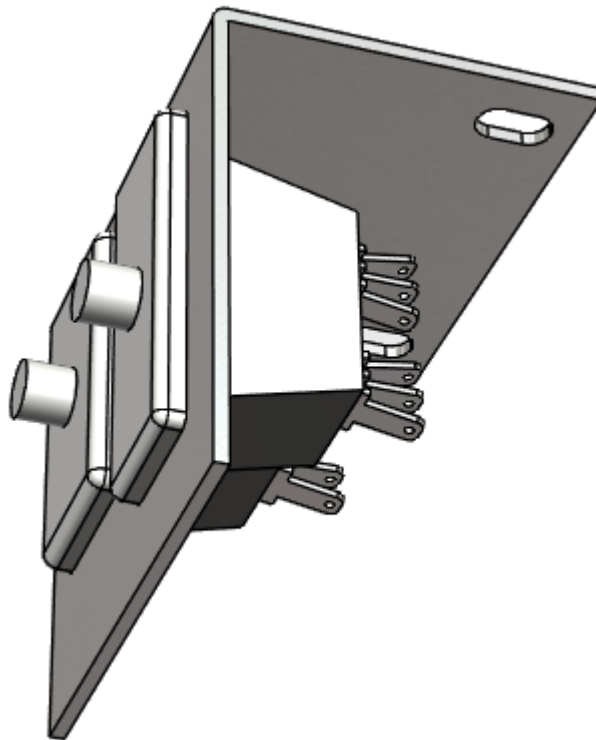
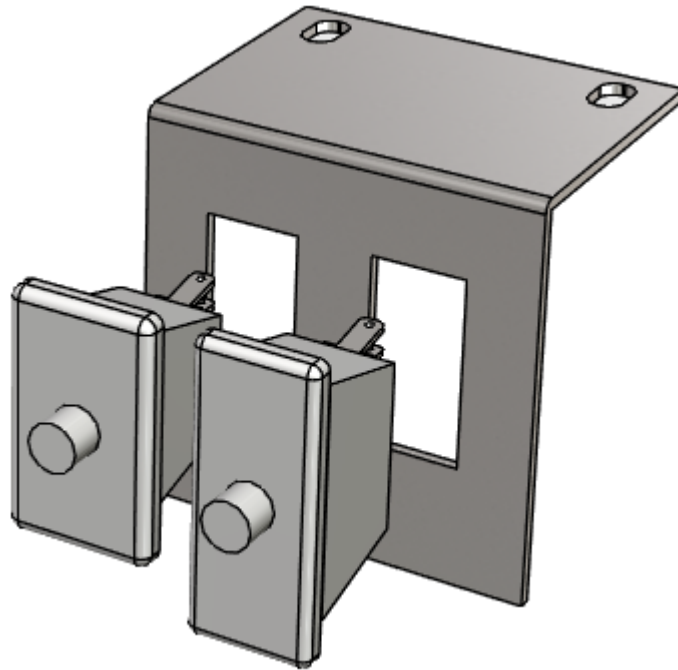
On a real machine, there's a switch that senses whether the coin door is open or closed. This is also useful to include on a virtual cab, because some of the emulated ROMs use it to control access to the operator menus. See Chapter 40, Coin Door for more.

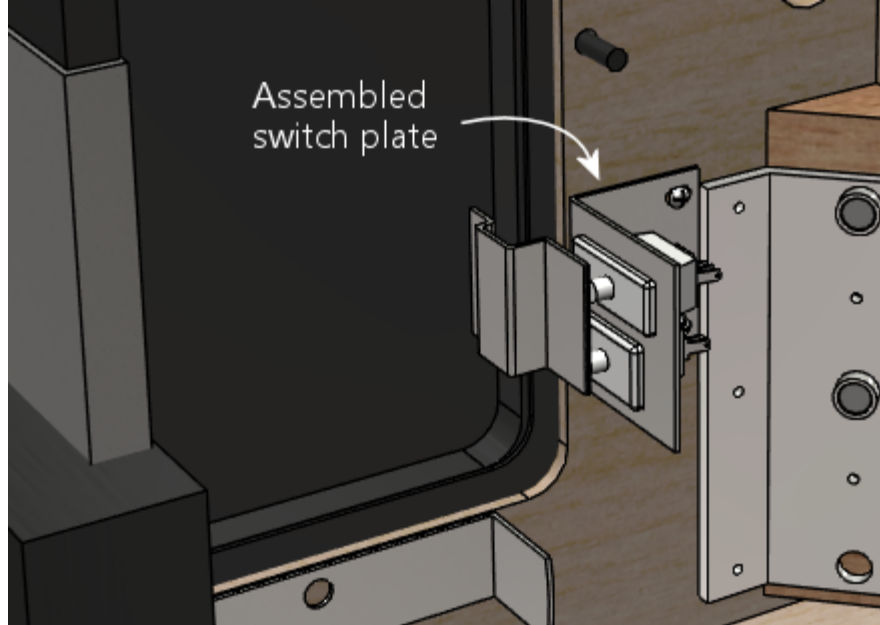
The coin door itself should have a pre-installed metal plate that acts as an actuator for the switch. This is positioned at the bottom of the door on the hinge side. It's attached to the door, so that it swings out when the door opens.

There are different ways to mount a coin door switch (which you can read more about in the Chapter 40, Coin Door chapter), but my recommendation is to use the standard pinball parts. They're purpose-built for this, so they're easy to install and reliable, and they're not particularly expensive. The standard parts consist of a metal mounting bracket and a "plunger" switch. The bracket is designed so that the plunger switch simply snaps - a couple of plastic clasps on the switch hold in place.

Snap the switch into the plate, then mount the plate so that actuator on the door presses the switch plunger all the way in when the door is closed. The plate mounts to the front wall of the cab with wood screws.

Note that the standard mounting plate has slots for two switches: a large switch with six connectors, and a small switch with three connectors. On the real machines, the large switch is used as an interlock to cut off high-voltage power to the playfield when the door is open, and the small switch is connected to the CPU to let the software know when the door is open. For a virtual cab, most people don't bother with the high-voltage interlock, since we don't tend to have any exposed high voltages to worry about in the first place. So you probably only need one switch, for the software. The large or small version will work equally well for that, so just install whichever one you bought and leave the other slot in the mounting plate empty.



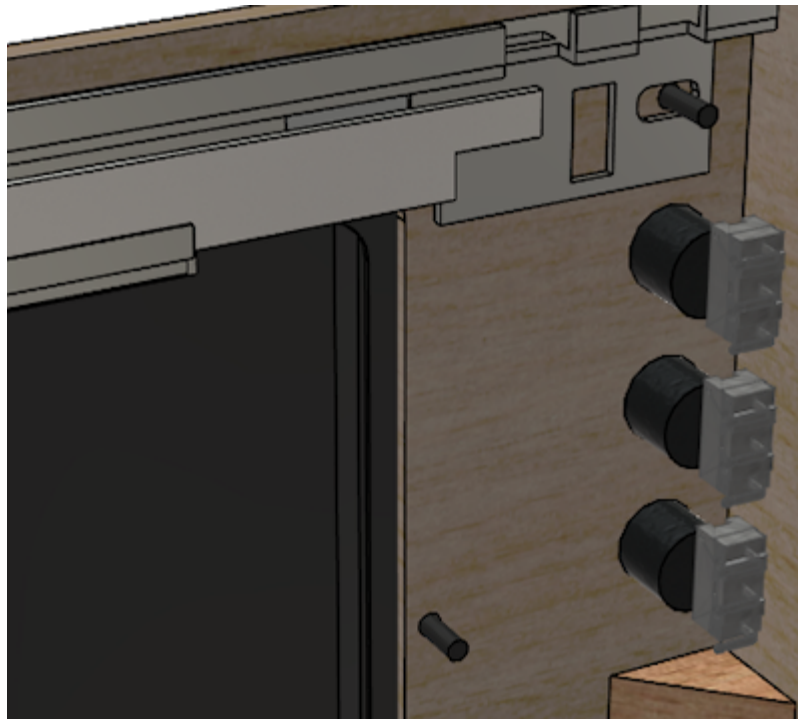
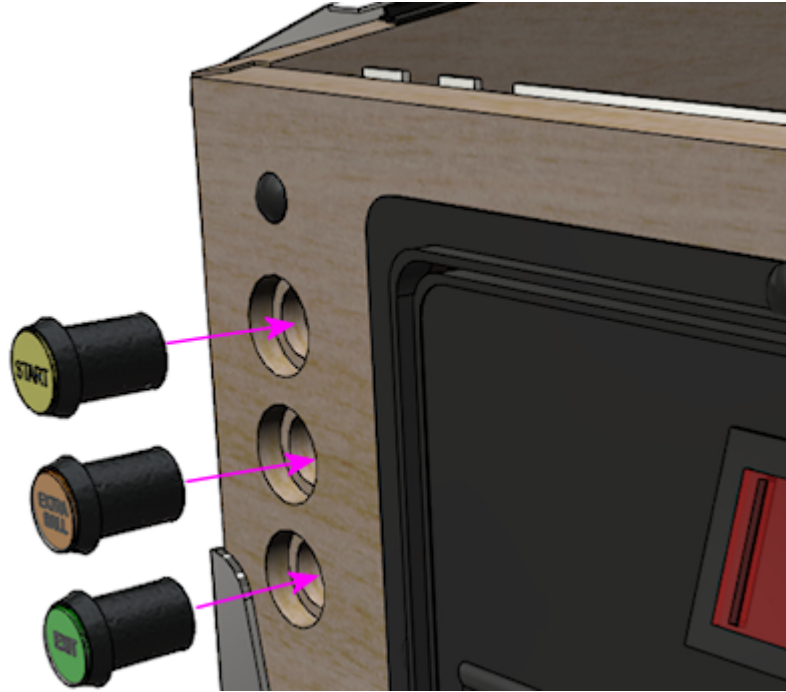


Front buttons

If you're using the common SuzoHapp "small round pushbutton" assemblies, they're easy to install. Start by disassembling the button. Gently twist the squarish base about 1/8 of a turn to free it, then pull it out. Unscrew the nut



Now just insert the button through the front wall hole (from the outside) and reverse the disassembly procedure: screw the nut back onto the shaft, and pop the lamp base assembly back into place, giving it a slight twist to lock it. The lamp base only fits in a certain orientation, so just rotate it until you find the magic spot.



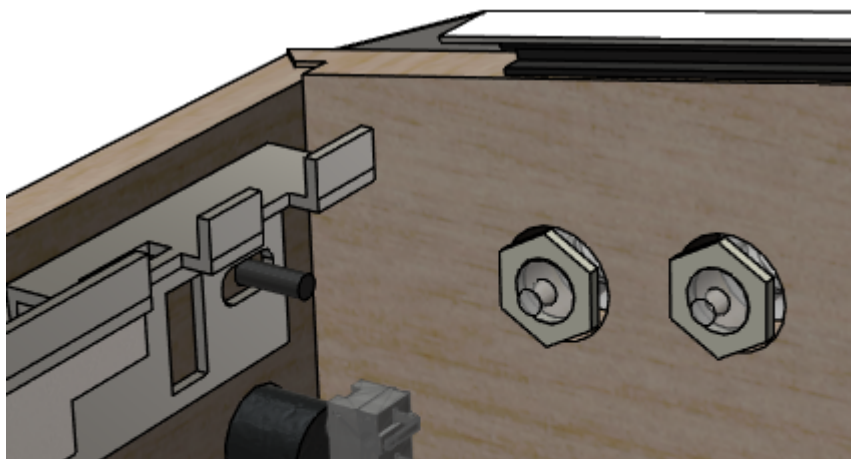
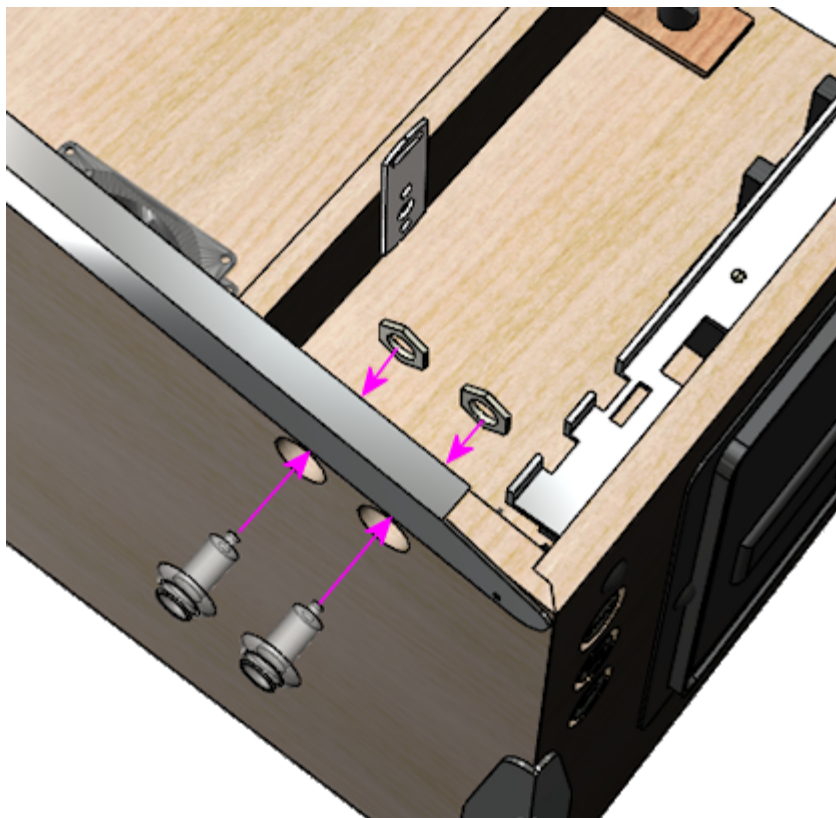
If you're installing a Launch Ball button, it works the same way.





Flipper buttons

The flipper buttons simply fit through the holes and are fastened with Palnuts on the inside. The rounded knob on the outside end of the button tends to be a tight squeeze - I guess that's intentional to keep them from getting wobbly over time. But it can take a little effort to force them into the hole the first time you install them. Seat them by applying pressure from the outside until the collars are flush with the cabinet wall. (I wouldn't try to force them flush by overtightening the Palnuts, since I'd be afraid of stripping the plastic threads.)



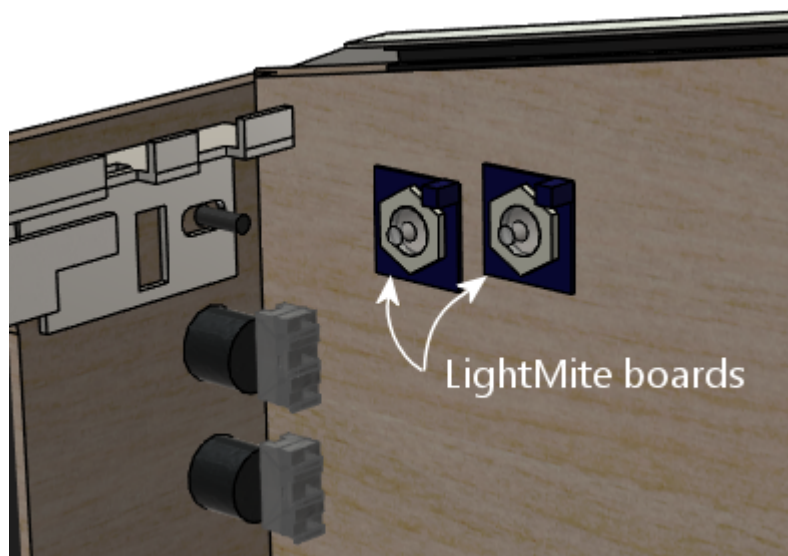


Note that if you drilled the flipper button holes straight through at $1\frac{1}{8}$ " (which is what I recommend), the Palnuts will be about the same size as the holes, so they won't clamp the buttons down very well. Don't worry - this will be fine as long as you're using one or both of the following:

- VirtuaPin leaf switch holders
- LightMite LED boards

If you're planning to install one of those, you can just leave the Palnuts loose for now and come back to this later. If you're not using one of those, and the Palnuts are too loose, you might need to add a suitable washer.

If you're installing the LightMite LED boards, they'll go under the Palnuts as illustrated below. You'll need to assemble them with LEDs and connectors first, so hold off on installing them if you haven't gotten to that yet. See Chapter 55, Button Lamps for more.



If you're installing the VirtuaPin leaf switch holders, they also install under the Palnut - it should be pretty obvious how those work.

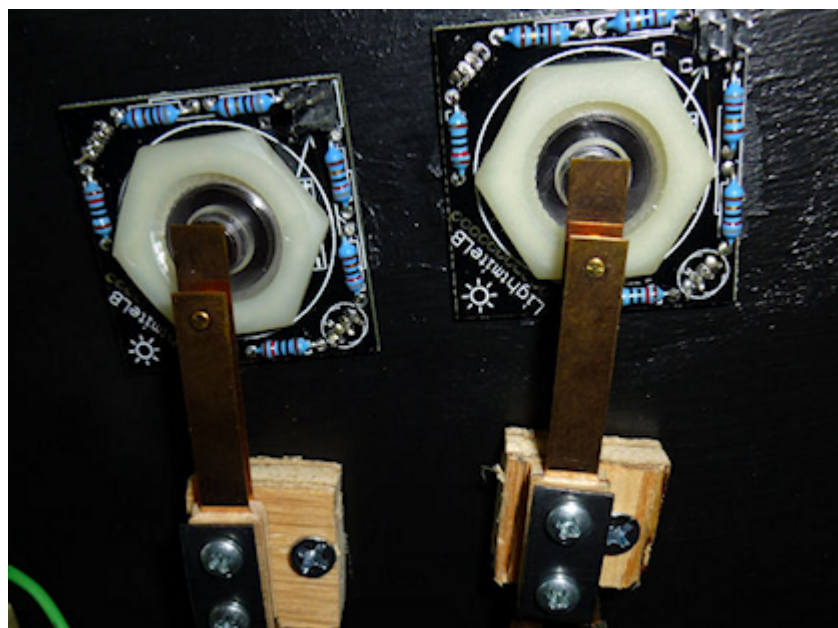
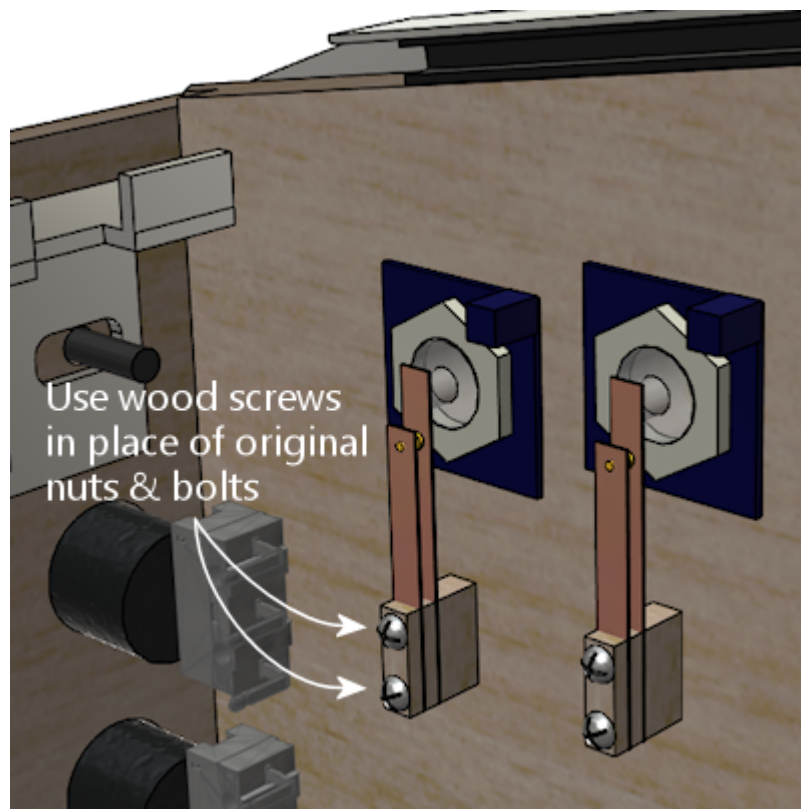
If you're not using the VirtuaPin leaf switch holders, you'll need to mount the leaf switches to the cab wall instead. This takes a tiny bit of improvisation.

Here's what I did. The standard leaf switches have little insulator plates at the bottom that separate the switch leaves. The whole thing is held together by a pair of bolts fastened with nuts. To attach these to the cab wall, you can take out the nuts and bolts and substitute wood screws. Use screws long enough to pass through the whole leaf switch assembly, with about $\frac{1}{2}$ " left over to screw into the cab wall.

That's *almost* all there is to it. But there's a slight snag: the switches will be too close to the cab wall if you mount them as-is. You need to add a little spacer to move them out from the wall about a quarter inch. I found that $\frac{3}{8}$ " plywood was just about right, so I cut some small (1" x 1") squares and used those as the spacers.

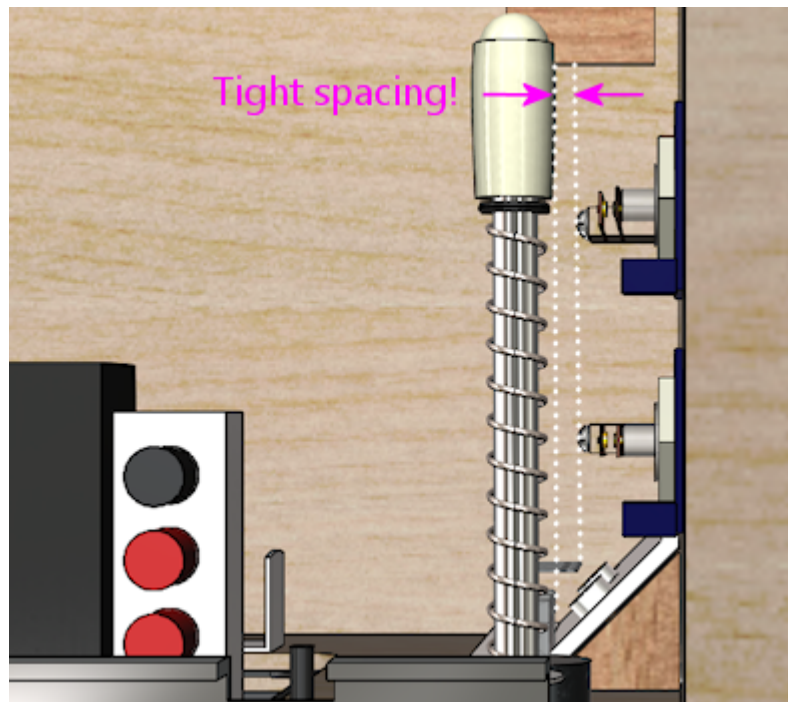
Remove nuts
and bolts

Add
Spacer





One last note before you actually install the switches. If you're installing a plunger, spacing on the plunger side will be tight. The flipper buttons happen to be positioned right alongside the plunger rod.



On the real machines, they leave just enough room to make it work, but we virtual people have an added challenge here, which is that we also need to install a plunger position sensor of some kind. That can add bulk around the plunger rod that isn't there on the real machines. All of the commercial and DIY sensor designers know this is an issue, and they take it into account in their designs, but space is so tight to begin with that some of the sensors push the limits here. So you might find it difficult to make everything fit.

There are two tricks that can help. The first is that you can mount the switches sideways or diagonally, instead of vertically as shown in the illustrations above. That can help get them out of the way of the moving plunger parts. I'd treat this as a last resort, since sideways mounts can create other conflicts (with the TV or apron, for example). The second trick only applies if you're using the VirtuaPin switch holders. If so, then your flipper buttons are extra-long, and you can swap them with shorter ones. The VirtuaPin switch holders only fit onto 1-3/8" buttons, whereas most modern commercial pinball machines use 1-1/8" buttons. So if you're using the longer buttons, you can save 1/4" by swapping them for the more common 1-1/8" buttons. The downside is that this requires ditching the VirtuaPin switch holders, which are convenient, and instead mounting the leaf switches to the cabinet wall as described above.

Adjusting the leaf switch gap

Most people in the pinball world agree that leaf switches are the only thing that feel right for flipper buttons, so they're almost obligatory in a virtual cab. But they do have one downside, which is that they sometimes need a little mechanical adjustment to get the switch blades aligned properly. Good operation depends on having just the right gap size between the contact points.

I wouldn't worry about making adjustments when first installing brand new leaf switches. I'd start with the assumption that they were aligned correctly at the factory. However, once you start using the buttons, keep an eye out for any flaky behavior: missed presses, random flipper flipping while holding a button down, weird auto-repeats, etc. If you see anything like that, you can take a closer look at the switches to see if they need adjustment. You might even have to re-adjust them from time to time, although in a home-use cab I wouldn't expect having to do that more than once every couple of years.

Whatever you do, **don't** clean the contacts with anything abrasive. You might see advice in "real pinball" contexts about sanding or scrubbing leaf switch contacts to remove oxidation. That's only for real pinball machines with high-voltage leaf switches, which use tungsten contact points. For a pin cab, it's better to use switches with gold contact points, since those work better for low voltages. Abrasive cleaning is bad for the gold contacts since it can remove the thin gold plating layer. The main reason that you see people recommend harsh scrubbing for the old tungsten switches is that tungsten oxidizes over time (especially in the presence of constant electrical switching), and the oxide layer is a good insulator, so you have to periodically scrape it off. Gold doesn't oxidize, so gold-contact switches don't tend to need much cleaning in the first place. But if you think your switches do need cleaning, use a slightly damp soft cloth and rub gently.

Testing: If you suspect flaky behavior from your leaf switches (or any other switches), but you're not sure, you can use the Pinscape Config Tool to take a closer look. (Assuming you're using Pinscape as your key encoder - if not, check your key encoder's instructions to see if it has a similar testing function.)

Fire up the Pinscape Config Tool, and click on the Button Tester icon on the main screen. This will bring up a window that gives you a direct view of each button switch at the hardware level. For the button or buttons that you suspect, press and hold the button and observe the status shown in the tester window. If the button is working properly, the on-screen status should show a nice, steady "On" indication, without any blinking or flickering. If you see the "On" indication flicker at all, you should try adjusting the leaf switch as described below. Likewise, when you release the button, the on-screen display should show a solid "Off" indication.

Tools: This is one of those jobs where you really need a special-purpose tool. The right tool makes this otherwise quite difficult job pretty easy. The right tool in this case is a "leaf switch wrench", which is essentially a little metal rod with a slit in one end that fits over a switch leaf and lets you bend the metal by a precise amount at a precise point. You can buy these from pinball vendors. On Pinball Life, search for "Ultimate Leaf Adjuster Tool". I bought one of those a while back for work on my real pinball machines, and I highly recommend it.

Dennis Miller on vpfforums sent me a great description of how he created his own leaf switch tool from scratch, so I'll pass that along in case you'd like to build one yourself as well:

All leaf bending needs to be done with the proper tool. I made mine out of 1/8" steel rod. I cut a slot 1/2" deep into the end of the rod with a hacksaw. I then heated and bent the rod at 90 degrees just above the slot so that the slot was almost parallel to the shaft. Slide the tool's slot over the leaf at its base insulator stack and bend very gently, a little at a time, to coax the leaf into position. The off-angle slot enables working close to cab walls.

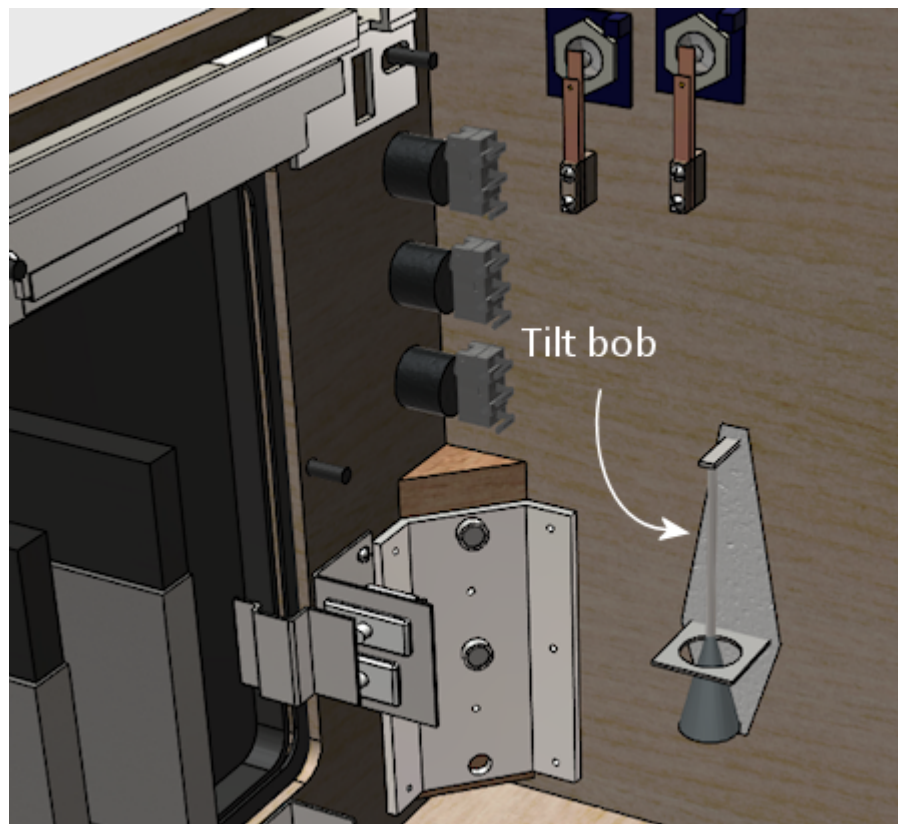
How to adjust: Approach this as an iterative process. Make small adjustments, test, and adjust again as needed. Make your bends towards the bottom of the leaves, close to the insulators.

- Start with the leaf on the button stem side. Adjust it so that it just touches the button stem when the button is at rest. There shouldn't be any open gap between the button stem and the leaf, so that the leaf starts moving the instant you start pressing the button. But don't overdo it; you don't want the leaf exerting too much extra pressure on the button, as that will make the button feel too stiff. The button already has its own spring for tensioning.
- Once the button side leaf is adjusted properly, adjust the other leaf so that the gap between the contact points is between 1/16" and 1/8".
- A 1/16" gap will make the button engage after pushing it in by about a quarter of its total travel. 1/8" is closer to the halfway point. I think the ideal point is a matter of taste, so test how it feels to see what you prefer.
- Once you've decided on the preferred gap size, you should adjust all of the flipper and MagnaSave buttons to use the same gap, to give them a consistent feel.

Tilt bob

The tilt bob conventionally goes at the front left corner of the cab. The exact placement isn't critical; just mount it in some free space below the left flipper buttons. Be sure to leave enough space that you'll be able to work on the wiring to the front buttons and coin door.

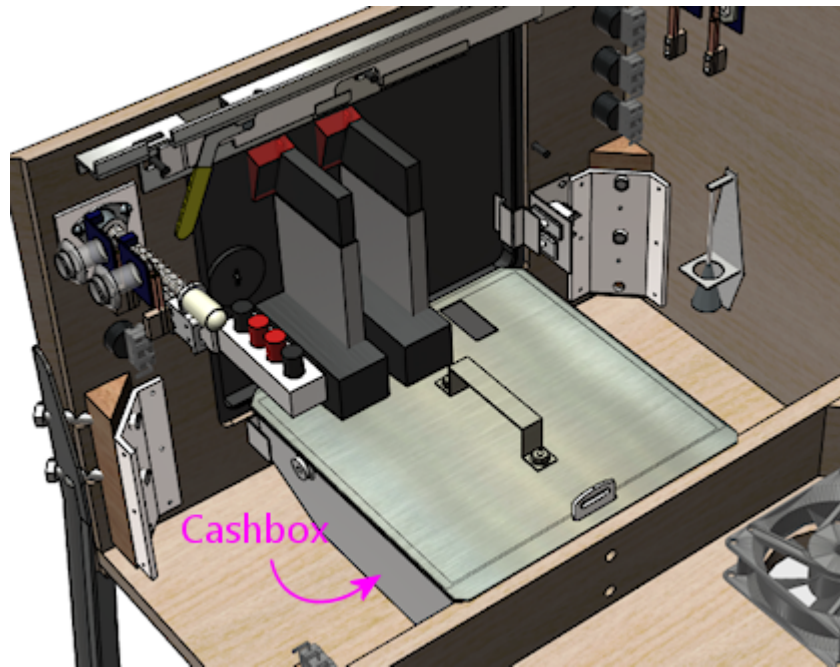
If you buy your tilt bob as a pre-assembled unit with its own mounting plate, mounting it is just a matter of screwing the mounting plate to the cab wall. It's almost as easy if you don't get the assembled version, though; you just have to mount the pendulum bracket and the contact ring separately, in the same arrangement as used in the pre-assembled units. See the illustration below.



Cashbox

This isn't something you have to "install", exactly; it just drops in. But the standard

type does take up a big chunk of space, so if you're using that, you might want to keep it in place (or keep it handy) while you're doing your space planning so that you take its bulky presence into account.



PC and PSU

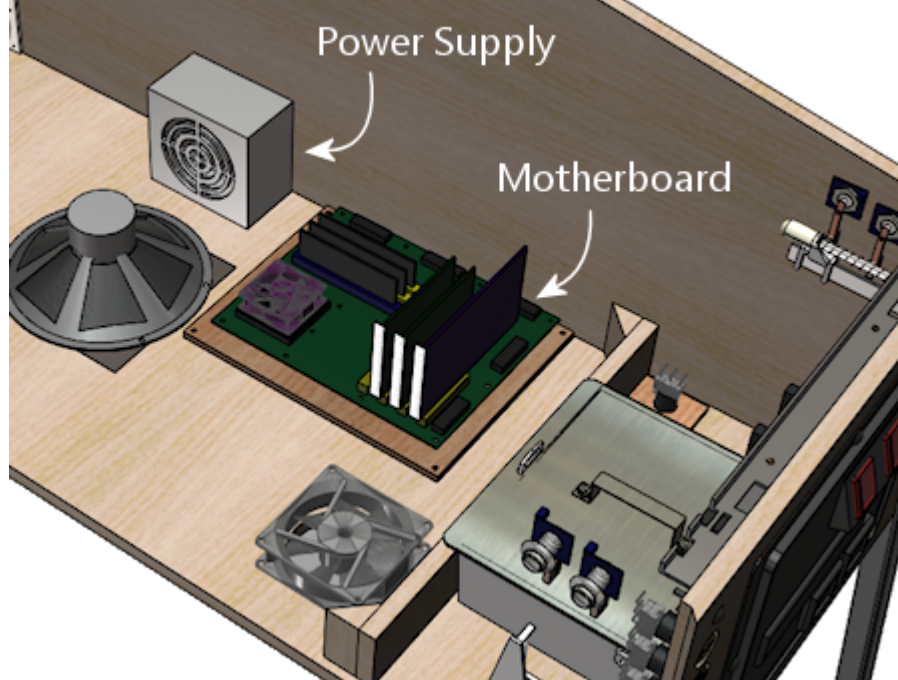
We're just about out of the standard "real pinball" parts, so let's turn to the virtual part of the system. I'd start with the PC, since it has a fairly large footprint.

Let's look at what we have available, now that we've taken into account most of the items that have to go at pre-determined locations:



Given this layout-so-far, there's an obvious place where something the size of an ATX motherboard or enclosed PC case would need to go:





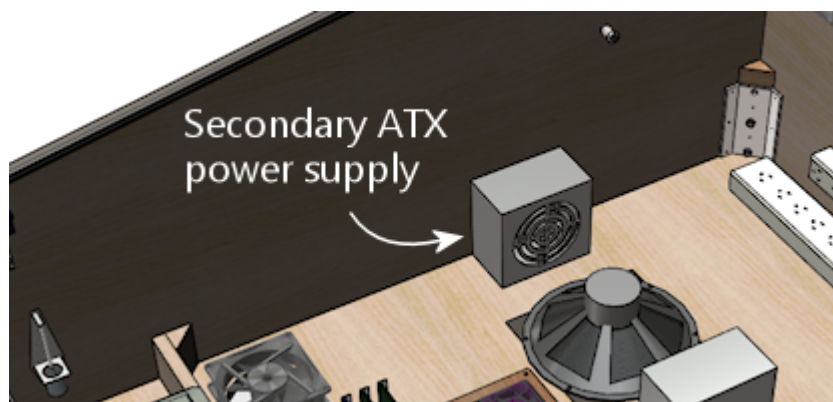
We have a little flexibility with the power supply, but only so much: it has to be close enough to the motherboard that the power cables for the motherboard and video card can reach their sockets. The obvious place is just behind the motherboard. That also happens to take good advantage of the space there, which is somewhat constrained by the presence of the subwoofer.

Setting up the PC hardware is a fairly significant project in itself, so we give that its own chapter, Chapter 27, *Installing the PC*. That section covers other ways of installing the PC components, such as enclosing them in a conventional desktop case, and goes into more detail about choosing a location and implementing the installation.

Secondary power supplies

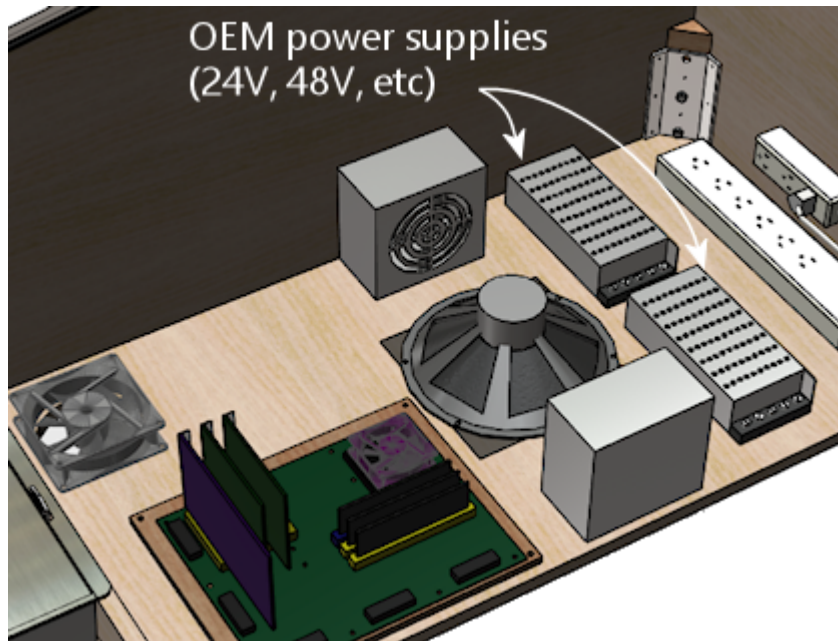
If you're installing feedback devices, you'll need to install power for them. More details can be found in Chapter 45, *Power Supplies for Feedback*, but the executive summary is that you can generally cover most of the bases with ATX power supply (that is, a separate unit of the same type used for the PC motherboard's power supply) and one or two generic OEM power supplies for higher voltages (such as 24V and/or 48V).

For the secondary ATX PSU, a good location is the mirror image of where we placed the PC power supply: on the other side of the subwoofer. Assuming you centered the subwoofer, there's a nice ATX PSU-sized space on either side, so we might as well use it that way.





The typical OEM power supplies come in long, low cases that fit well into the space remaining at the back of the cabinet, between the subwoofer and the power strips.



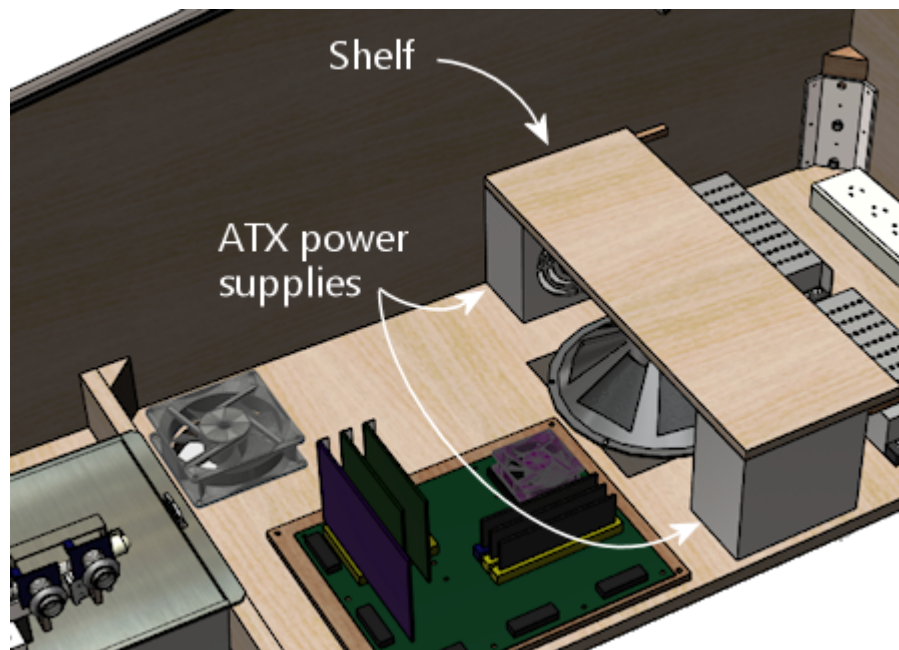
The OEM supplies are usually a good physical fit for this space, and they're also a good functional fit, in light of what I said earlier about how the back section becomes increasingly inconvenient to work in as you build out the cabinet. The power supplies are a good set-it-and-forget-it kind of thing for a hard-to-access space. They don't have any controls; you just plug them into power.

You *do* have to be able to access their power outputs, though, whenever you want to plug in a new device. So there's a bit of advance planning you should do when you install them. Specifically, you should wire their outputs to connectors located somewhere more accessible in the cabinet, more towards the front. Many people set up a group of terminal strips like the one illustrated below somewhere readily accessible, one for each voltage level, so that they can easily connect each new device to its appropriate supply when the time comes. (Be sure to protect any exposed terminals like these with plastic covers, so that loose wires don't accidentally inject high voltages into unsuspecting logic boards.)



A nice side benefit of installing the two ATX power supplies across the aisle from one another is that we can use them to construct a little shelf across the width of the cabinet. That'll be useful later: you can see that the floor space is already almost all

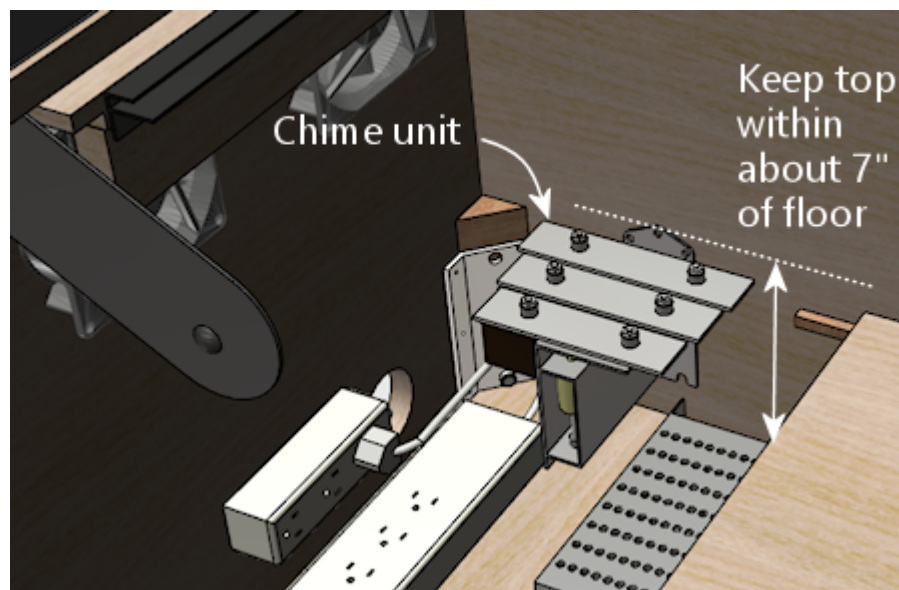
gone, and we still have a number of important things left to find room for.

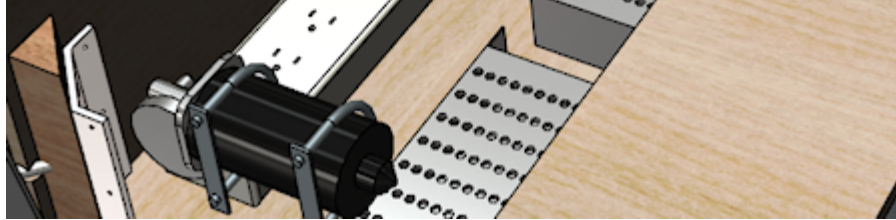


If you've been paying attention, you know how important I think it is that you be able to access everything in the cabinet even after it's fully assembled - the principle I call serviceability. So you should be sure that this shelf can be easily removed! Don't glue it in or anything like that. At the very least, fasten it with a couple of easily removable screws. But better yet, use something you can undo without tools: attach it to the power supplies with Velcro, for example, or use toggle latches to lock it down. That way it'll only take a few seconds to remove it if you have to get to the power supplies.

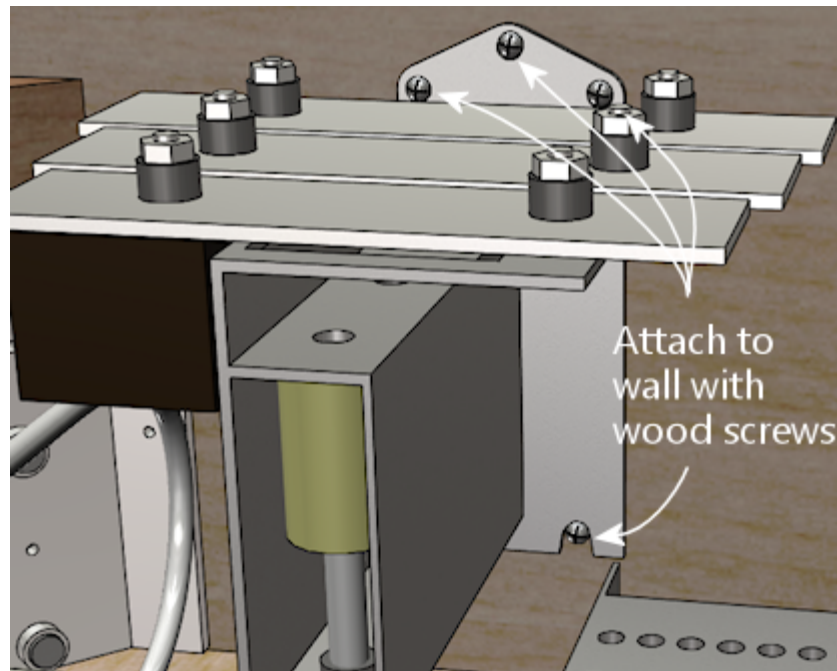
Chime unit

See Chapter 64, Chimes and Bells. This is a little percussion instrument that replicates the iconic bings and bongs of the electro-mechanical pinballs from the 1960s and 70s. The best way I know to accurately reproduce the original sound is to find an authentic used chime unit from an old machine, as they have some engineering that's hard to replicate in a DIY design. The real units are quite bulky, though, which limits where we can put them. The only place where a chime unit will fit in our hypothetical fully-loaded cab is in a corner at the back.





The original chime units are designed to be mounted to a side wall. Use wood screws to attach it via the integrated mounting plate.

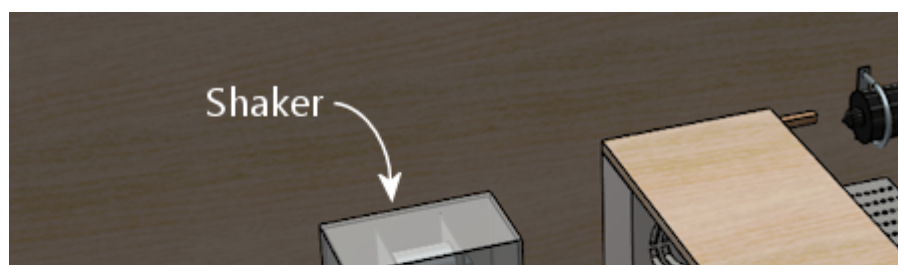


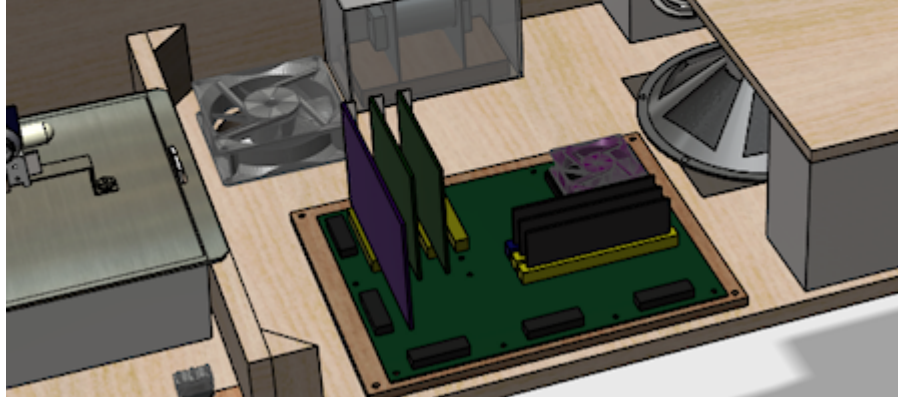
Try to keep the top within about 7" of the floor. This will help avoid any clearance issues with the back of the TV when you lift it up. (Assuming you opt for a liftable TV mounting, as outlined in Chapter 29, Playfield TV Mounting.)

Much as I don't like hiding things away in the back of the machine, we really don't have much choice when it comes to the chime unit. There's just not enough space anywhere else. If your cab won't be as fully loaded as the one we're developing here, though, you might have some space for it in a more convenient area, so by all means put it somewhere better if possible. I don't think the placement makes any significant difference acoustically. For what it's worth, most of the original machines that used these units also placed them in a corner - typically the right front corner, below the plunger.

Shaker

See Chapter 61, Shaker motors. The shaker is another bulky toy, and in this case it **must** be mounted on the cabinet floor to get the proper effect. Fortunately, we have one large floor section still remaining, mid-cab, opposite the PC motherboard.





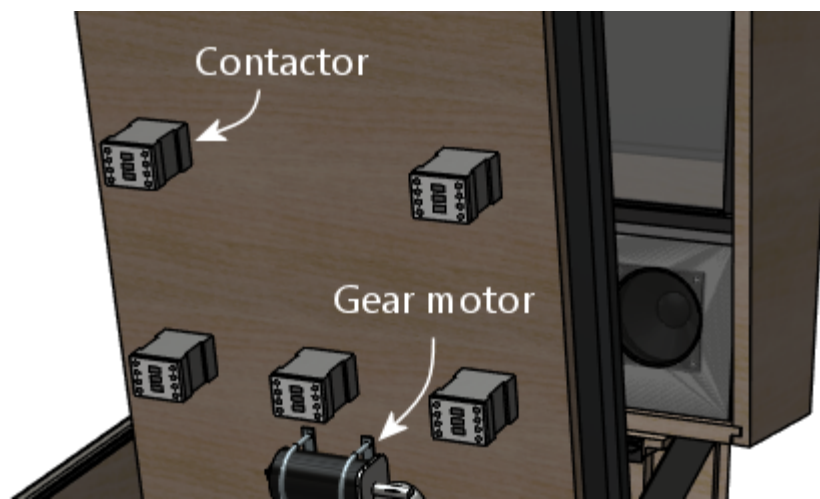
Happily, this works out well, as this is just about exactly where we'd put the shaker anyway, to get the best tactile effect, if space were no concern. You want the shaker to be mounted with its motor axis parallel to the cabinet's long axis, and that's a perfect fit for the available space. You also want the shaker to be in roughly the middle of the cab front-to-back so that it imparts a balanced sideways motion. There's no benefit in centering it side-to-side, so I'd mount near the wall, to leave more room around the PC for cable connections.

Gear motor

See Chapter 62, Gear motors. These are meant to reproduce the sound of the motorized playfield features on many pinballs from the 1980s and 1990s, such as Thing from *The Addams Family* or the castle gate from *Medieval Madness*. To localize the sound effect properly, the gear motor should be somewhere towards the back of the cabinet, since the playfield features it's meant to imitate are typically towards the back of the playfield. (The playfield features in question are all unique to each game, so they're all in their own unique locations, but for the most part they're somewhere near the center rear of the playfield.)

There are two rather different types of motors that pin cab builders tend to use for these. One type is the small robotics servomotors you can buy on eBay. Those are so compact that space planning really isn't an issue for them. The other popular type is an automotive windshield wiper motor. Those are quite a lot larger, and do require that you block out some space for them. We'll proceed with the assumption that you're working with the larger type and need to find a place for it.

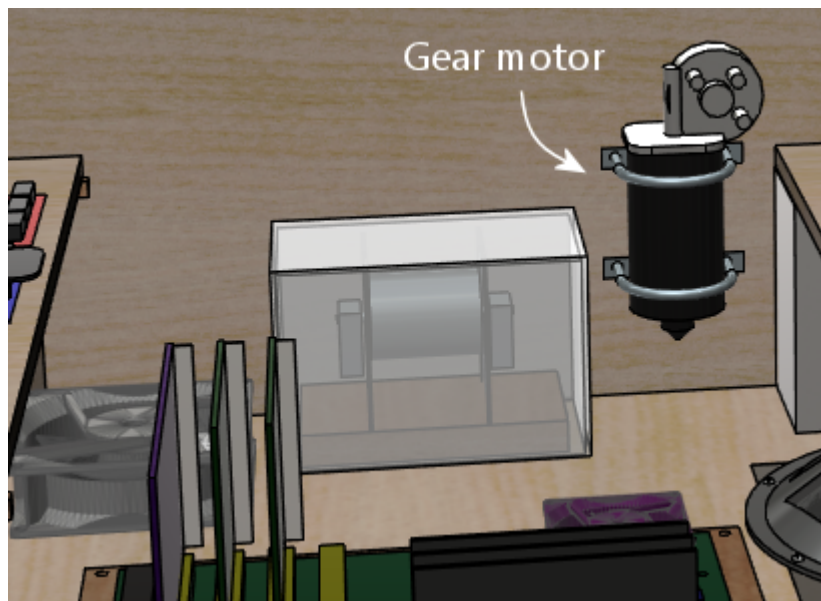
If you're using a liftable TV frame design, you might be able to mount the gear motor on the bottom side of the TV frame. That would let you put it right in the middle of the playfield area, which is the ideal location for the sound effect, plus it's easy to access for service. This is the right option if you have a compatible TV mounting.



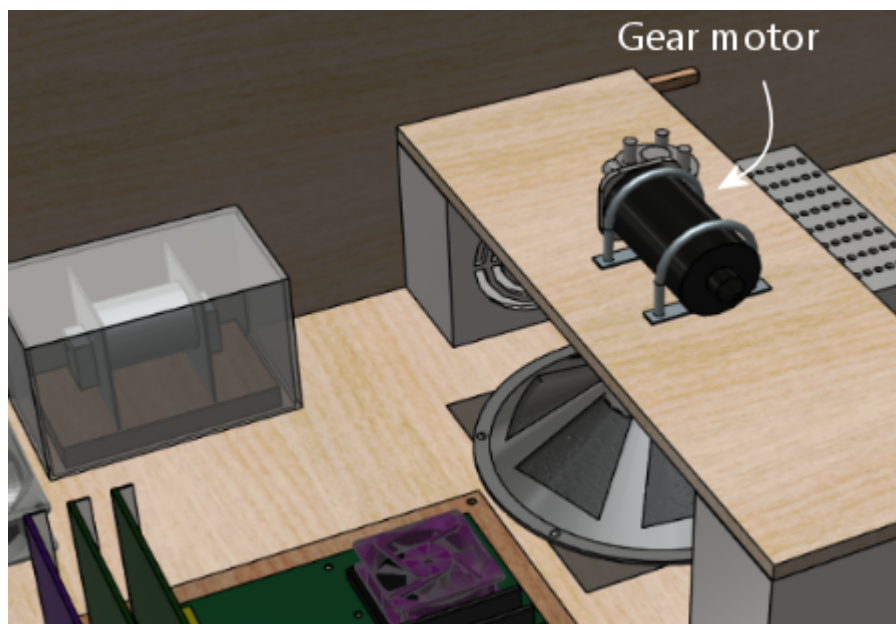


In the illustration above, we're assuming that the contactors for the bumpers and slingshots are also mounted under the TV. A gear motor should fit nicely between the "bumper" rows in the back half of the playfield. See "Mounting contactors under the TV" below for more about this.

If an under-the-TV mounting doesn't work in your cab, there are several places it might fit. One possibility is to place it alongside the shaker:

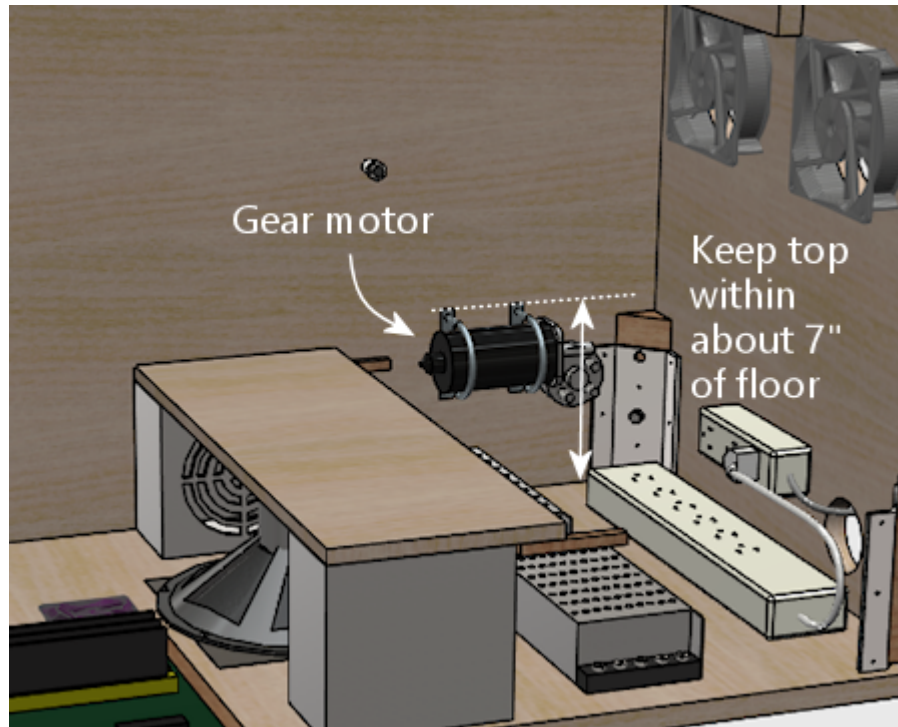


A second option is to use the little shelf we built over the ATX power supply and subwoofer area:



The shelf is probably the best location in terms of localizing the audio effect, and it's a great location in terms of service access. The only problem is that there are a couple of other devices we'll come to later that we'll need the space for. So we're *not* going to be able to leave it here in the model we're developing, but you can keep this location in mind as an option in your cab, if the space ends up being available after you consider where the rest of the parts go.

A third option is to place it in a corner at the back:



If you go this route, try to keep the top within about 7" of the floor. This will help avoid any clearance issues with the back of the TV when you lift it up. (Assuming you opt for a liftable TV mounting, as outlined in Chapter 29, Playfield TV Mounting.)

As I've said a few times, this isn't a great area to mount just about anything, because it's hard to reach into in an assembled cab. But gear motors tend to be zero-maintenance, so if you have to put something back here, a gear motor isn't the worst choice. What I'd recommend is to use a mounting apparatus that you can remove without tools if necessary. Something like this, perhaps:

- Mount the motor to a small sheet of plywood (cut just large enough for the job) with a pair of "U" clamps, which you can buy at any hardware store
- Use something like a Z-clip (a heavy-duty type of picture hanger) to hang the plywood carrier on the wall
- Secure the bottom with a thumb screw or toggle latch, so that it can't come loose from the hanger

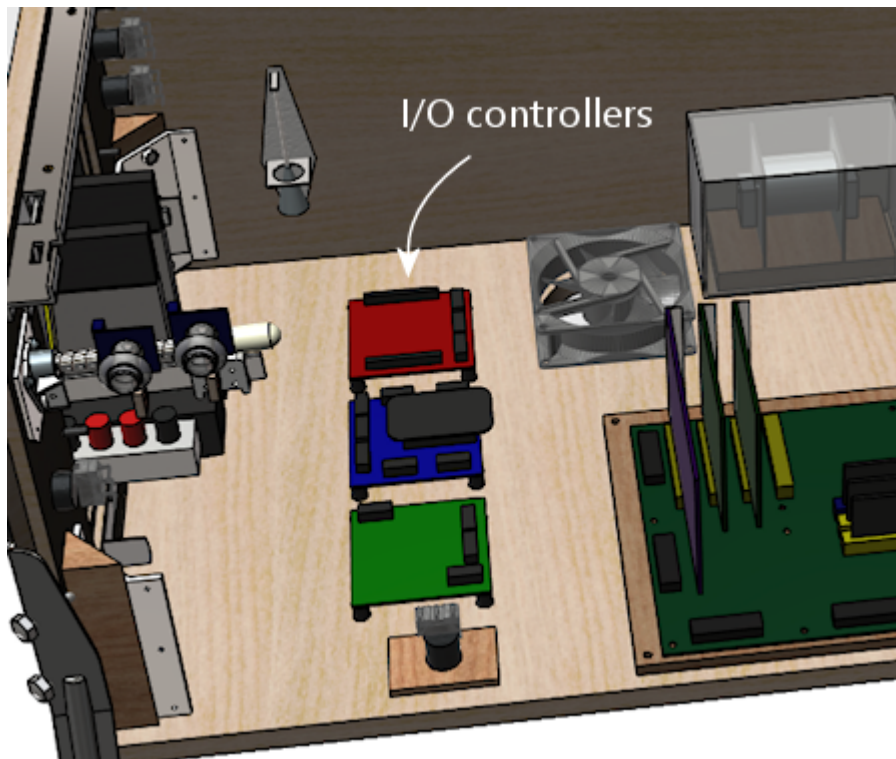
If you do need to access the gear motor, this will let you take it out as a unit without having to do anything too complicated in the confined space. Once it's out, you can make whatever changes are needed, and just as easily put the whole unit back in place.

Controllers

See Chapter 12, I/O Controllers. A pin cab requires some special USB devices to connect the button inputs, plunger sensor, and feedback devices. There are several options for these, but whichever you choose, you're going to have some little circuit boards that you'll need to mount somewhere in the cab. Most cabs need two or three boards, most of which are on the order of 4" by 4".

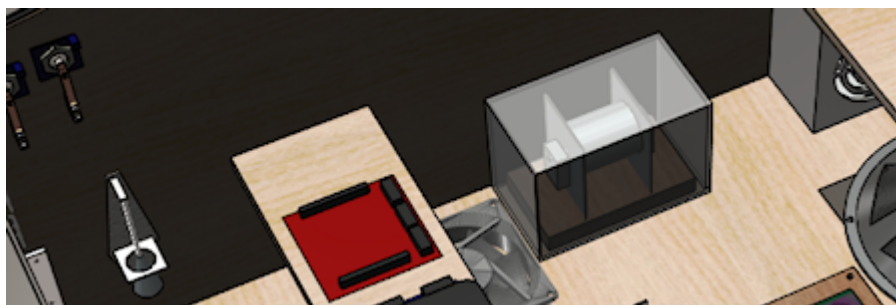
Most of these boards can go just about anywhere that's convenient, but there's one type of board that's pretty particular about location: the accelerometer, also known as the nudge sensor. That board should be mounted horizontally, close to the front of the cab, preferably close to the center of the cab. The accelerometer senses the cabinet's motion, and it does the best job at that if it's mounted in a central location near the front.

If you're *not* using a full-sized cashbox, then you still have a nice open space at the front, where the cashbox would go on a real machine. That's an ideal spot for the controllers.



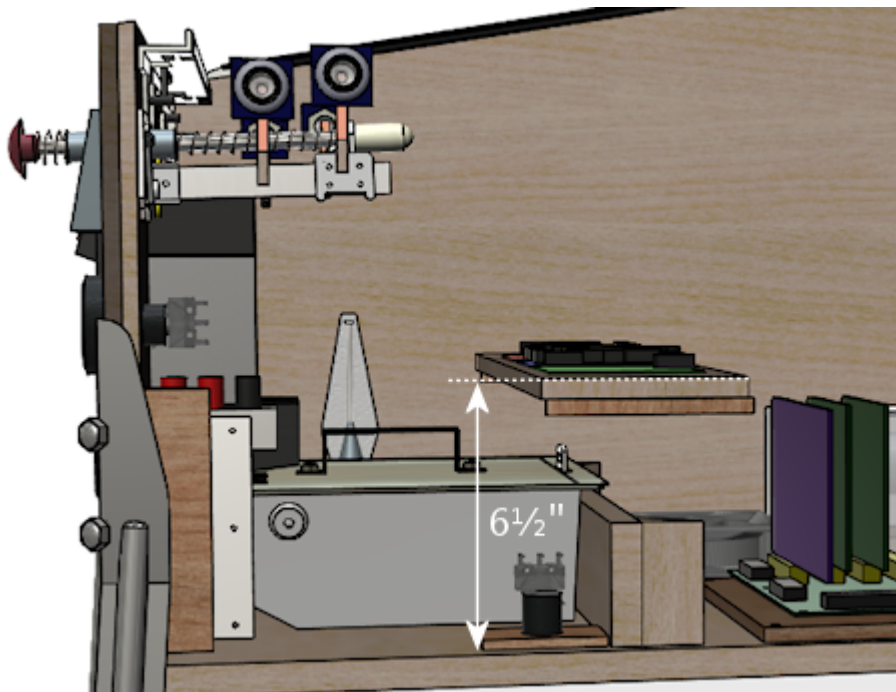
If you *are* using a full-sized cashbox, we're in a bit of a jam now, because there's not enough floor space left for the controllers. This is, in fact, why I don't have a real cashbox in my own cab. But I don't really like my hokey improvised substitute (a plastic food container that happens to be about the right height, with holes cut in the lid to line up with the coin slots). So given that this section is about an idealized ultimate cab with everything, let's see how we could make this work.

My proposal is basically to create some new floor space, by thinking three-dimensionally:





What you're looking at is a shelf, running the width of the cabinet, about 6½" above the floor, positioned over the back portion of the cashbox.



This reclaims the floor space that we gave up to the cashbox. It's at the right position for our accelerometer, and it gives us enough space to mount a typical complement of I/O controller boards.

Some important considerations:

- For the sake of the accelerometer, the shelf must be quite solid, and quite solidly mounted to the cabinet. It **must** move with the cabinet; it shouldn't impart any extra vibration or wobble of its own. For this reason, I think this shelf needs to be securely screwed in, not held down with Velcro or anything like that. But I think it's okay for this shelf to be more or less permanent, since, if it's properly positioned, it won't block access to anything.
- Position the shelf so that it doesn't block access to the motherboard. This is especially important given that it needs to be so solidly (permanently) attached to the cabinet.
- Use a sturdy material. I'd recommend a good quality ¾" hardwood plywood, the same sort of material used for the cabinet itself. The shelf doesn't have to support any significant amount of weight, but remember that we want it to be very solid so that we get good accelerometer readings.

- At the recommended height, the shelf will leave enough space that you can still conveniently maneuver the cashbox in and out through the coin door, as intended.
- At this height, the shelf should also leave comfortable clearance for a typical playfield TV with my recommended mounting. For the purposes of the model, I assumed what are probably the worst-case conditions in terms of how much headroom we have here: a fairly thick TV (3.5") and a "deep" mounting style (with the TV at full playfield depth). With those assumptions, we still have about 3" of headroom to work with here. That's plenty of space for any of the controllers I've encountered.



Fuses

See Chapter 84, Fuses. Fuses can be used to protect your output controller from overloads. You don't necessarily have to include a fuse for every device, but it's good to cover the higher power devices, such as motors and solenoids.

Your output controller might have its own built-in fuse holders, but most of them don't, so fuses usually have to be installed separately. We're going to assume you're installing them separately.

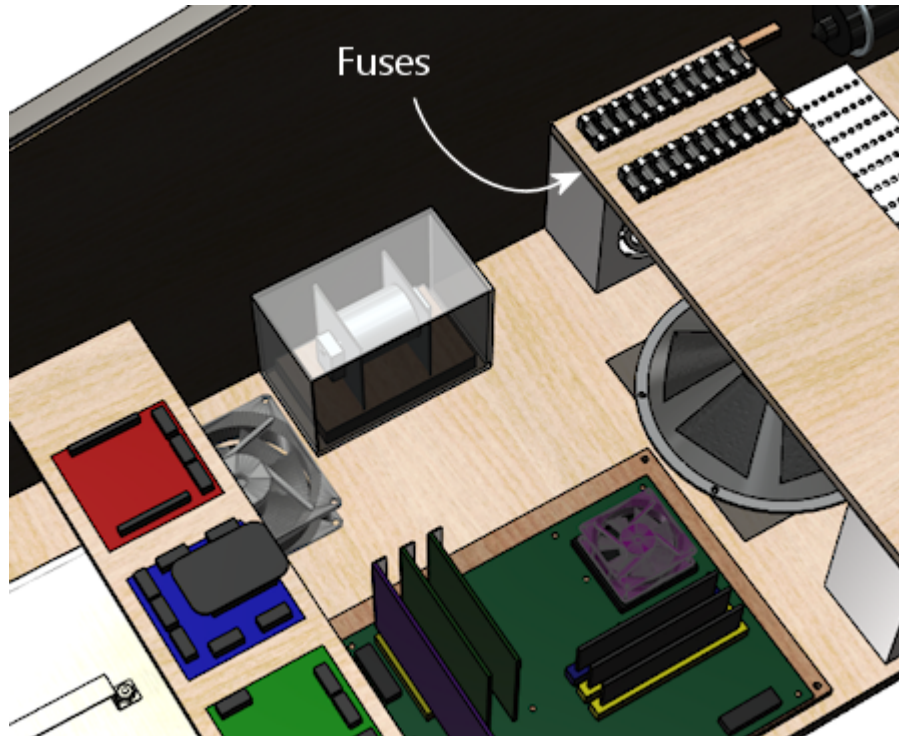
There are many types of fuses and fuse holders. For my own cab, I went with the type that's common on the real pinball machines (not for the sake of realism, but just because it saved me the trouble of researching all of the other options). Those are the so-called 3AG glass cartridge fuses, which look like this:



These can be used with little plastic holders that look like this:



This type of holder is designed to be mounted to any sort of surface with a screw (which you can see in the photo), so we can mount these on any convenient wood surface on the cab, such as the floor, a wall, or that center shelf we created earlier over the ATX power supplies and subwoofer:



I like the idea of centralizing the fuses in one big set like this, since it makes it easier to find the fuse for a given circuit. However, it has some disadvantages: it takes up a big block of space, and it requires extra runs of wire to and from the central fuse panel. You also have to make a chart of what each fuse is connected to.

Another option that you might prefer is to place each fuse near the device it's connected to. The individual fuse holders are small and can mount just about anywhere, so ad hoc placement per fuse avoids the need to allocate space for a central fuse panel. And it can save a lot of wire, since you can place each fuse somewhere along the section of wire that you'd have to run out to the device anyway. Finally, it might be easier to figure out which fuse goes with which device this way, as long as you can manage to place each fuse physically close to its device.

Keep in mind that the type of fuse holder pictured above has two exposed metal terminals. If you're creating a central fuse panel out of these, you should consider placing a plastic cover over it to protect it from accidental contact from tools or loose wires. If you're scattering the fuses (rather than creating a central panel), you might want to use a more fully enclosed fuse holder instead of the open type. For example, take a look at the Littlefuse 155 series in-line twist-lock holders. Those are designed so that there are no exposed terminals.

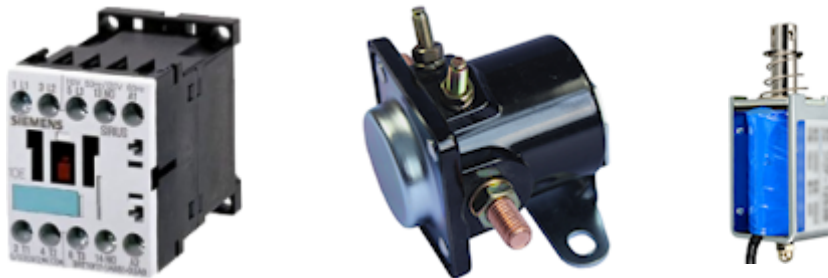


Littlefuse 155 series in-line 3AG fuse holders

Contactors (and other solenoid simulators)

See Chapter 59, Flippers, Bumpers, and Slingshots. The real pinball machines have a lot of powerful solenoids that kick the ball around and actuate other playfield mechanisms. They're are strong enough that you can not only hear them but feel the kick. Virtual pinball software can manage the audio part with recorded audio, although that tends to be a weak imitation that you'd never mistake for the real thing. In a pin cab, we can do better, by simulating the kick of the solenoids with actual solenoids. That can get a lot closer to the real sound, and can also reproduce the tactile effect.

There are several types of solenoid-based devices that pin cab users employ as substitutes for pinball solenoids: contactors (such as the Siemens type pictured below), automotive starter relays, generic open-frame solenoids, and even real pinball solenoids and their associated mechanisms.



Common devices used to simulate pinball coil effects: Siemens contactors; Ford starter relays; generic open-frame solenoids.

For the purposes of our illustrations, we'll use the Seimens contactors. The Ford starter relays and most open-frame relays should comfortably fit the same spaces, so you should be able to substitute them without making other changes.

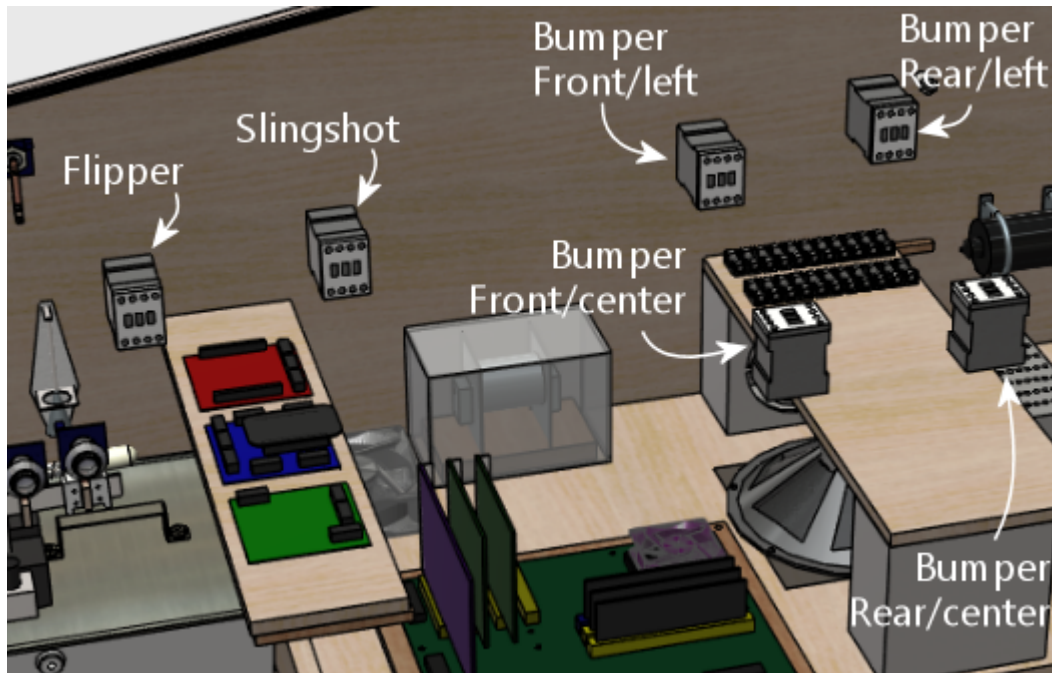
The standard complement of contactors consists of 10 units:

- Two flippers (left and right)
- Two slingshots (left and right)
- Six bumpers (three across the middle, three across the back)

The goal is to locate each contactor so that it matches up with the position of the device it's intended to simulate, as it'll appear on the main TV screen when you're

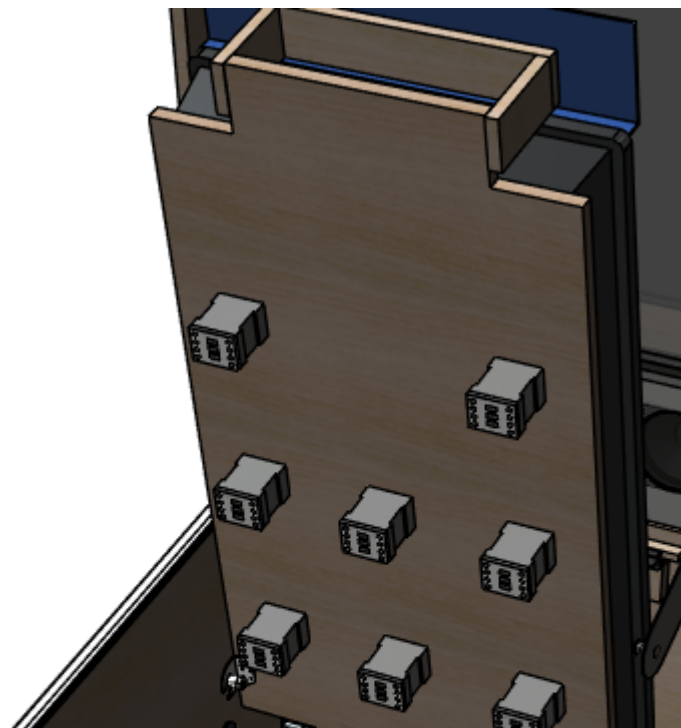
playing a game. So you want the contactor that's going to serve as the "left flipper" to line up roughly with where the simulated left flipper is drawn on the TV screen. It's obviously impossible to get that perfect for every game when you're going to have hundreds of simulated games to choose from. But all pinball playfields tend to follow the same template for the core elements around the flipper area, and anyway, we don't have to get it perfect, just close enough to be convincing.

So, taking the desired positions into mind, here's how we can arrange the ten devices to fit the available space. Note that the devices illustrated on the left wall are mirrored on the right wall, but we're leaving them out of the diagram for the sake of readability.



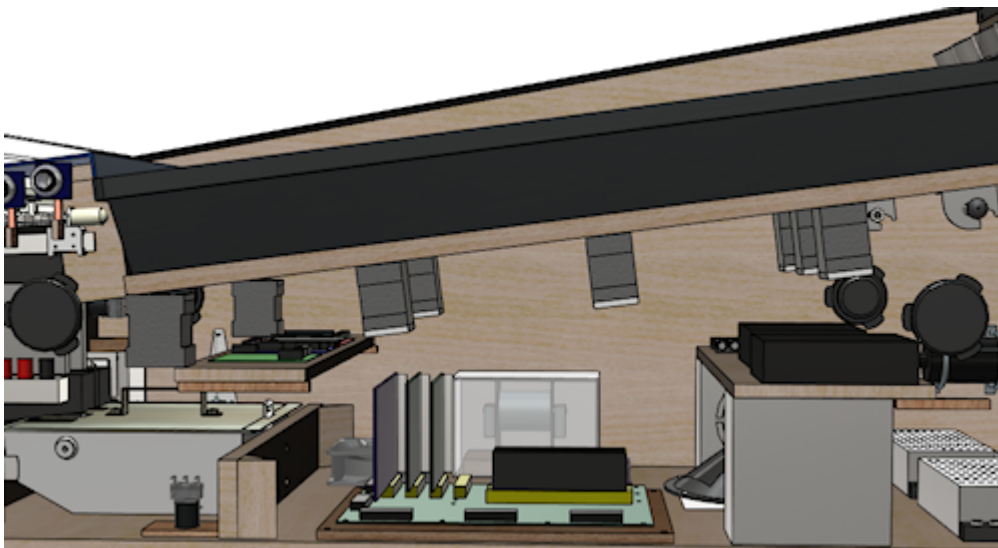
Mounting contactors under the TV

If you're using a liftable TV mounting like the one described in Chapter 29, Playfield TV Mounting, you can move most of these to the bottom side of the TV mounting frame, as illustrated below.





Here we've moved all of the contactors except the flippers to the underside of the TV frame. We left the flippers where they were (on the side walls), because the natural place for them on the TV frame is a bit too close to the shelf where the I/O controllers are located. If we didn't have that shelf, we could easily move the flipper contactors to the TV frame as well. This is what it looks like when we lower the TV back into its normal position:



As you can see, there's lots of room for everything, except for the area around the I/O controller shelf.

The under-TV mounting style has some distinct advantages:

- It places the devices closer to the on-screen elements they're intended to simulate
- It frees up space along the cabinet walls
- There's more room for larger devices than the original side-wall mounting
- The contactors are easier to access for service, since they're more out in the open after you lift the TV up

I don't think there are any real disadvantages, either. I think it's the right way to go if you have a suitable TV mounting. And it's practically required if you plan to use real pinball mechanisms for any of the solenoid devices; they're too large to be workable with the side-wall mounting.

There is one important consideration if you go this route. You'll definitely need to use a pluggable connector for the wiring to the contactors, so that you can remove the whole TV-and-frame assembly from the cab without having to cut wires. I recommend using one of the Molex .062" wire-to-wire connectors, which are available in plugs with up to 12 pins. That lets bundle the wiring for the whole set of

contactors into a single plug.

In-cab speakers

See Chapter 41, Audio Systems. We've already covered the subwoofer, which traditionally goes on the floor in the middle of the cab. But many cabs also include a set of mid-range speakers inside the main body. These are usually *in addition to* the speakers in the backbox, and serve a different purpose. The backbox speakers are there to play the ROM music and voice effects. The speakers in the cab are there to reproduce mechanical sounds that aren't already covered by the solenoids and contactors. For example, the sound of the ball rolling and bumping into things.

Visual Pinball has the ability to separate the music from the mechanical effects and play each type of through a separate set of speakers. Playing back the mechanical effects through speakers inside the cabinet makes them seem to come from the playfield, improving the illusion. VP 10 takes this one step further, by supporting a four-speaker "surround sound" arrangement that localizes each sound effect to the right point in the playfield plane.

There are several ways to configure in-cabinet speakers. If I were building a new cab today, the only option I'd consider would be a four-exciter system. This takes advantage of VP 10's spatial localization capability to position effects in different parts of the playfield.

An "exciter", by the way, is a type of speaker that works by making the surface it's attached to vibrate. A conventional speaker works by vibrating a paper cone. An exciter uses whatever it's attached to in place of the paper cone. They're better than conventional speakers for an in-cab speaker system for several reasons:

- They're much smaller than regular speakers, so it's easier to find room for them in a crowded cab
- They're made specifically to be mounted to flat surfaces (like the wall of a pin cab!)
- They work by transmitting their sound energy through whatever they're attached to, which better reproduces the way mechanical sounds in a real pinball machine travel through the cabinet
- Transmitting the sound through the cab wall produces more of a tactile effect than a regular speaker does

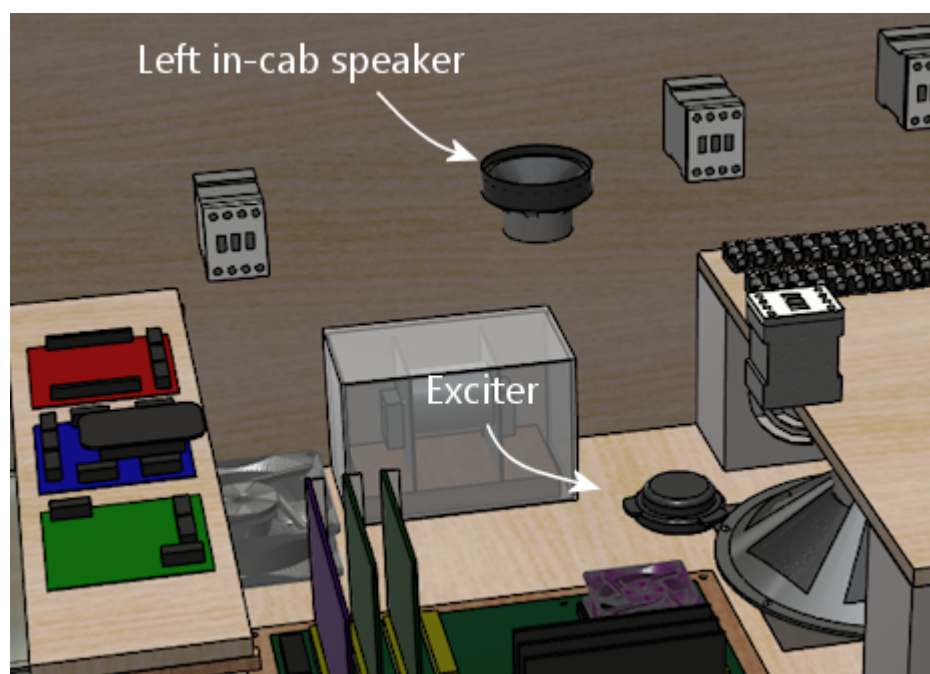
The ideal mounting positions are at roughly the corners of the TV. That's the arrangement that the VP software assumes when it calculates the volume mixing levels to create the illusion that the sound is coming from a particular point in space. It's pretty simple to install exciters this way: just install two on each side wall, below the TV, one near the front and one near the back. Most exciters are quite flat and compact, so it's not hard to find room even with everything we've installed so far.





Some people add one or two subwoofers to this setup as well. I personally don't think that's necessary. For the types of sound effects we're talking about in a pin cab, the only reason you'd want a subwoofer is for more of a tactile effect. Exciters are already good at producing tactile effects because of the way they transmit the sound energy through the cab wall, so I think a subwoofer is redundant. Besides, if you really need more bass from these channels, you can make Windows mix the low-frequency bands from the surround channels into the main subwoofer output.

I built my own cab before VP supported the four-channel surround system, so I took a simpler approach, with two regular speakers and tactile subwoofer:



This produces a decent effect, certainly better than no in-cab speakers at all, but the lack of spatial positioning is sometimes too obvious. It's particularly noticeable when the ball is near the top or bottom of the playfield, since the sound always comes from a fixed spot in the middle. That's why I'd go with the four-speaker system now that it's an option.

Amplifiers

See Chapter 41, Audio Systems. Unless you're using powered speakers, you'll need some amplifiers. Most pin cab builders use small car amps, since they're compact and (like everything automotive) run on 12VDC power. Most of the cheap units can power a stereo speaker pair or a stereo pair plus subwoofer (the latter being known as a 2.1-channel amp). Many higher-end car amps can power four independent channels.

You'll typically need the following:

- One 2.1 amp for the backbox speakers + main subwoofer
- A second 2-channel or 2.1 amp for the front surround speakers

- A third 2-channel or 2.1 amp for the rear surround speakers

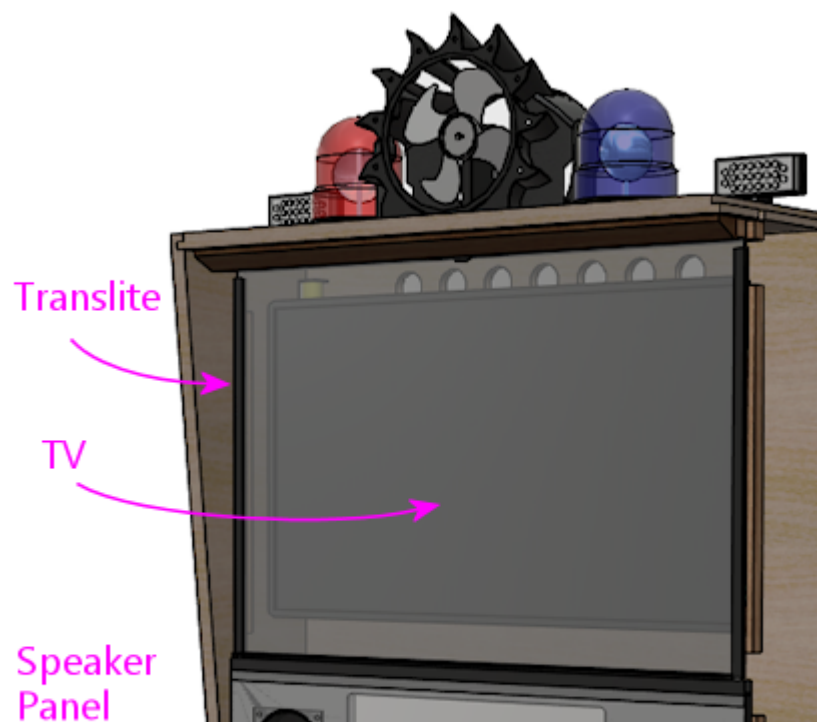
I've been keeping space for a couple of these units open on the shelf over the subwoofer area:

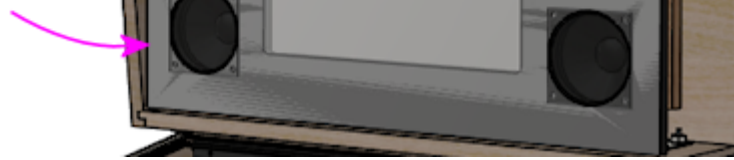


What you can fit here will obviously depend on the specific equipment you choose. You can probably fit two small car amps, and maybe three, if you're able to stack two of them vertically.

Backbox

In the backbox, as in the main cabinet, we have a top layer that's visible to the player. In the backbox this consists of the translite, backglass TV, and speaker/DMD panel.

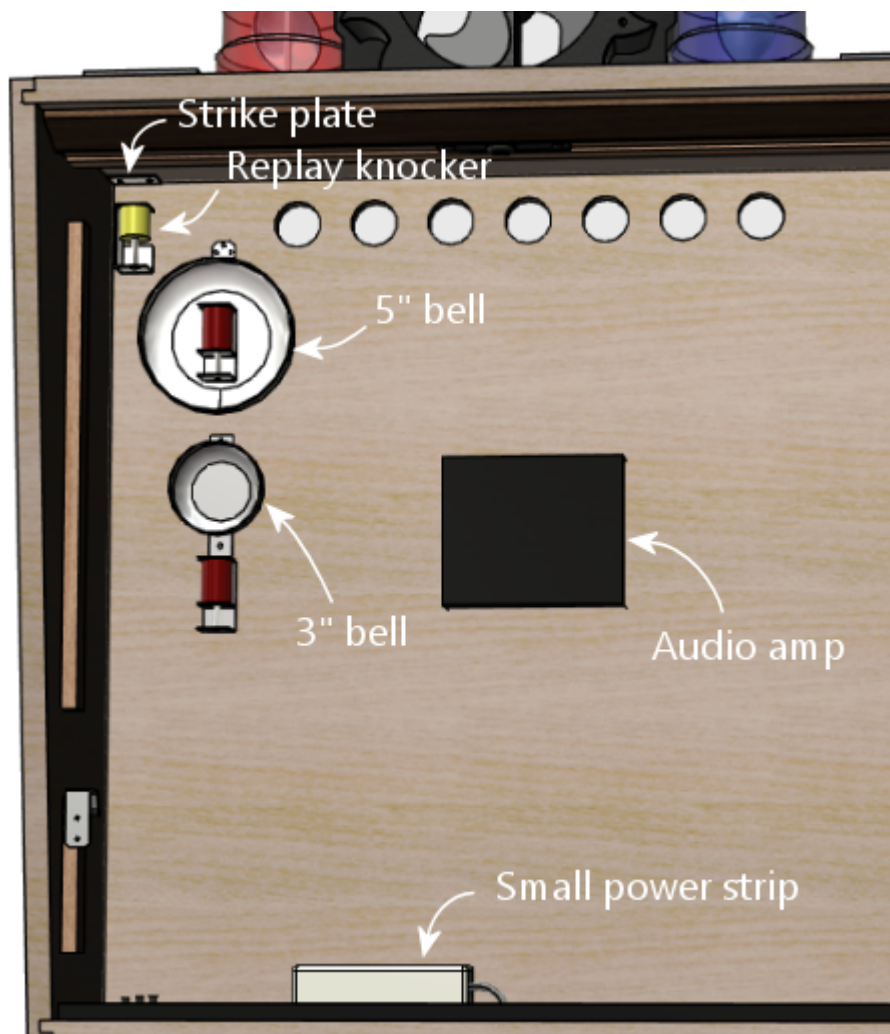




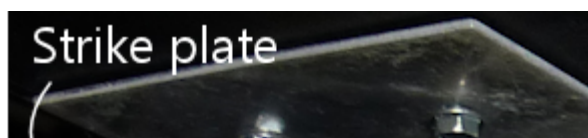
Those are all covered in detail in other sections:

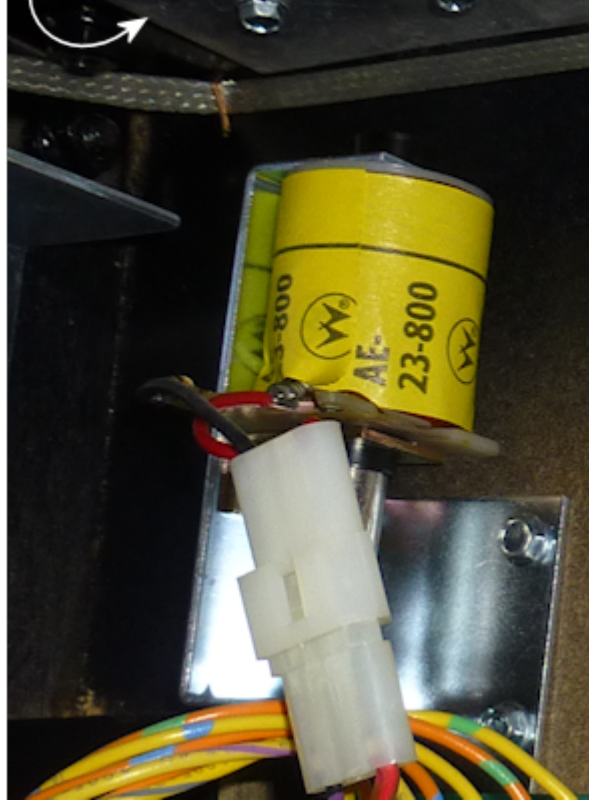
- "Creating a translite" in Chapter 24, Backbox Hardware Installation
- Chapter 8, Selecting a Backbox TV (designing the backbox layout and choosing a backbox TV)
- Chapter 30, Backbox TV Mounting (installing the TV)
- Chapter 31, Speaker/DMD Panel (fabricating and assembling the speaker/DMD panel)

There are some additional items that we can fit into the backbox, mounted on the back wall.



- Replay knocker: typically mounted at the top of the backbox in a corner. The knocker coil is mounted so that the open end points up at the ceiling, with about a 1" gap to the ceiling. The metal strike plate is mounted on the ceiling right above it. See Chapter 60, Replay Knockers.





- Shell bells. If you're a big fan of machines from the electro-mechanical era, you can install a couple of round bells with solenoid hammers. Similar bells were used in many machines from the 1960s and 70s. These serve exactly the same function as chimes, so in a way they're redundant with a chime unit, but the reason you might want to have both is that bells and chimes each have their own distinctive sound. Some games from the EM era had bells and others had chimes, so you can more accurately re-create a greater variety of games if you have both. The backbox is a good place to install bells if you have them; the bells have a large footprint, but they're flat enough to fit behind a TV (in most cases, anyway), so this takes good advantage of the wide but shallow space in the backbox. As a nice bonus, it's true to the originals: it's where bells were usually situated in the EM machines. See Chapter 64, Chimes and Bells.
- Repeating bell (not shown). The shell bells above work like chimes in that they fire with one hammer strike at a time. There's a different kind of bell used on some machines from the 1980s, which rings continuously when energized, like an old-fashioned alarm clock or telephone ringer. These look just like the shell bells, so they're an equally good fit for this space. There should be plenty of room to add one of these if desired. See Chapter 64, Chimes and Bells.
- Audio amplifier. We already proposed a place where you can fit a couple of car-radio amplifiers into the main cabinet. You might also be able to fit an amplifier into the backbox, either as an alternative to the main-cabinet mounting or in addition (which might be necessary if you need four channels of audio in the main cab for a surround-sound setup). You'll probably have about 1" to 2" of depth to work with behind the TV, which is enough to fit a small amplifier.
- Power strip. There should be enough space on the floor of the backbox behind the DMD panel to install a small power strip. A 3-outlet strip fit easily in my backbox in this area. It's convenient to have a few outlets here, so that you can plug in the backbox items (TV, DMD panel, audio amp) without having to run more cables through to the main cabinet.

27. Installing the PC

This section is about physically installing the PC components in your cabinet. It's not about building the PC per se - things like how to plug RAM into the motherboard, connect disks, install the graphics card, connect the power supply cables, and so on. That sort of thing varies depending on the exact mix of parts you're using, so that's more the domain of the instructions that came with your components. What we're going to talk about here is how all of the PC parts fit into the pin cab.

Location

For a lot of our cabinet design questions, we can follow the example of the real pinball machines, rather than having to invent everything ourselves. But the real machines don't have PCs inside! So they don't give us anything to go on for where to put the computer.

It might seem at first glance that there's at least a close parallel in the real machines. The solid-state machines from the 1980s and later might not have PCs inside, but they do contain computers of a sort. They have a bunch of circuit boards that control the game (including CPUs and RAM, even), located in the backbox. So if we *could* follow what the real machines do, we'd put the PC in the backbox.

But that doesn't translate very well to virtual cabs. The space requirements of the respective "computers" are too different. In the real machines, the electronics were all purpose-built custom boards, so the designers were able to build them specifically to fit into the available space in the backbox. We don't have that option; we have to work with standard PC motherboards, which are designed to fit in a standard PC case, not the more confined space of a backbox. When you take into account that we also have to fit a TV into the backbox, we only have about 1" of depth to work with - which wouldn't even be enough room for a CPU fan, let alone a video card.

With the backbox ruled out, that leaves the main cabinet. That's assuming you want the project to be self-contained, anyway, which is what most cab builders want. If that's not a requirement for you, it opens up the additional option of keeping the PC in a separate, external case, and placing it on the floor.

Fortunately, the main cabinet works nicely as the location for the PC. It's a large space, and it's mostly empty in a real machine, so we don't have to give up any standard real-pinball elements to make space available for the PC. And even though we're adding a TV to the main cabinet, we actually get more space to work with than in a real machine, because we don't have a physical playfield to contend with. We just have the TV. A modern flat-screen TV is much thinner than a typical pinball playfield, because playfields always have a bunch of big solenoid coils and other mechanical parts sticking out from the bottom side. So we come out a little ahead of the real machines in terms of usable interior space.

Enclosure options

There are several different approaches to installing the PC. Here are the ones I've seen on the forums:

- Build the PC in a conventional case, and put the case inside the cab
- Build the PC in a conventional case, and put the case *outside* the cab, such as on the floor next to it
- Use an "open-frame" case or "motherboard tray", which has the backbone of a

regular case for attaching the motherboard, expansion cards, disk drives, and power supply, but doesn't have an enclosure

- Mount the individual PC parts directly in the cabinet, with no case or frame of any kind

I think the last option (no case) is probably the most common. It's how I set up my own cab, and it worked well for me. But each approach has its own merits. I wouldn't say that there's any single way that's best for everyone, or conversely that any of them are flat-out wrong. So let's look at each one in detail, to help you decide what will work best for your cab, and give you some ideas about how to implement your chosen approach.

Conventional case, inside the cab

This is the most straightforward approach: build your PC in a conventional desktop or mid-tower case, and then stick the case inside the pin cab.



Typical PC mid-tower case, 16" x 16" x 7", lying on its side in the cab just behind the cashbox area.

The big question to ask when considering this approach is whether or not it'll fit. If you're building a full-sized cab, and you're using a typical ATX desktop case or mid-tower case, it should work, although it'll be a little tight. A typical mid-tower ATX case is in the neighborhood of 16" x 16" x 7", and a standard-body WPC cabinet has a floor space of 20.5" x 50", so there's room for the case's footprint no matter how you orient it. Vertical space is tighter. The cabinet interior is about 14" high at the front and about 21" high at the back, but remember that also have to fit a TV into the same space. What's left over after installing the TV will depend on exactly where you position the TV. Some people like to put it at the very top edge of the cab, millimeters from the glass, but most people like to set it further in, at about the depth of a real pinball playfield. The latter case is the more difficult one in terms of space left for the PC, so to be conservative, that's the one we'll assume. If the TV is about 4" thick, a playfield-level mounting will leave about 7" of vertical clearance at the very front - just enough for the 16x16x7-inch case lying on its side. But that's

only at the very front; you get more headroom the further back you put the PC, since the TV slopes up. And you'll actually want to put the PC back a little further anyway, to leave room at the front for the coin door and plunger. So the answer is yes, it'll fit, at least for this hypothetical 16x16x7 case.

Advantages of this approach:

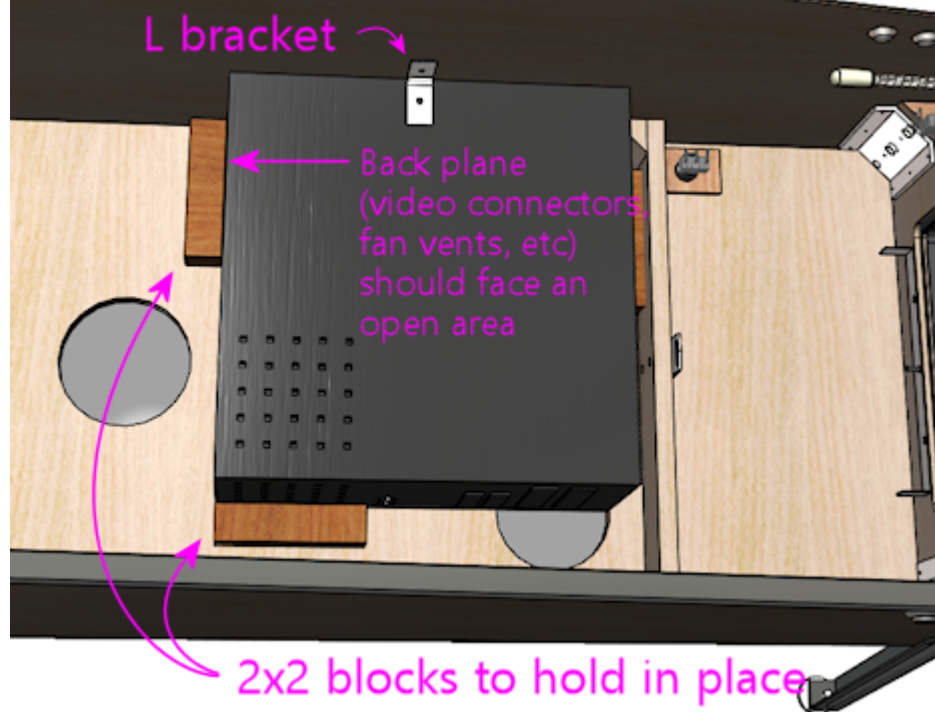
- It's easy to build the PC, since you build it the "normal" way
- It's self-contained
- You can easily remove the whole thing to work on the PC outside of the cab
- The case protects the PC components from loose objects in the cab

Disadvantages:

- It takes up a lot of space; you'll have less space room for other things (such as feedback devices) than with an open-frame case or no case
- Putting an enclosed case inside an enclosed pinball cabinet might make cooling less effective

Considerations if you choose this approach:

- When building the main cabinet, you should consider moving the subwoofer opening further towards the back than the location shown in our WPC-style plans. (And note that our plans already place it further back than it was in the real WPC machines, where it was nearly centered front-to-back.) The mock-up diagram above uses the subwoofer placement from our plans, and you can see that there's almost no room between the subwoofer opening and the PC case. Keep in mind that the speaker itself will be slightly larger in diameter than the opening.
- The case will need to be oriented to leave enough clearance for video cables, USB cables, and power cords. That means that the back plane of the case can't be flush with a wall. This might affect where the fan intake vent ends up, so figure it out early so that you can take it into account when build the cabinet.
- Position the case so that its air intake and cooling fan exhaust vents aren't blocked. Ideally, position the intake vent close to an opening in the cab floor, so that the intake can draw in outside air directly (rather than the warmer recirculated air within the cab).
- You probably won't be able to fit a separate intake fan with this arrangement. That's okay as long as the PC case intake is near a floor vent opening. The PC's case fan will pull air in through the vent and thus will serve the same function that an intake fan would have served.
- If air flow does turn out to be a problem, you can always take the cover off the case and just use the chassis, similar to the "open-frame" case style discussed below. (That gives up one of the advantages of using a full case, though: the protection afforded by the complete enclosure.)
- Secure the case in place so that it doesn't slide around the cabinet. I'd probably use wood blocks (perhaps nominal 2x2 strips) at the edges, and something like an L-bracket to lock it down vertically. I'd want the top bracket to be easily removable so that you can take the PC out when needed. One of the big advantages I see in using a full case is that it makes it easy to remove the PC as a unit to work on it outside of the cab. Perhaps fasten the L-bracket with a thumbscrew or wing bolt, so that you can remove it without tools, or use something more of the nature of a toggle latch.



- You'll need to connect wiring to the PC's soft power button so that you can turn the PC on with the cabinet's power button. (See Chapter 11, Power Switching.) I'd find the wires leading to the case power button, and splice my own extra wires onto these, running the extra wires outside the case through an available opening. You can then connect these to your cabinet power button.

Remember that we want to make it easy to remove the whole PC case for maintenance work, so be sure to use some kind of modular, pluggable connector for the switch wiring. Don't hard-wire it. When you want to remove the case, you can just unplug that connector to free the wires. See Chapter 80, Connectors.

Open-frame case

An "open-frame" PC case, also sometimes called a motherboard tray, is an interesting compromise between using a full conventional PC case and no case at all.

An open-frame case is basically just the backbone of a regular PC case, without any enclosure. It has the usual mounting apparatus for the motherboard, power supply, disks, and expansion cards, in the usual spatial arrangement. Installing all of the parts works just like in a regular case, except that there's no cover to put on when you're done. These cases are sold mostly for people doing test builds and experimentation, where they want easy access for frequent component changes.

Pros:

- Assembling the PC components is as easy as for a conventional case
- Slightly more compact than an enclosed case
- Open air flow for cooling

Cons:

- Still fairly large, even though it's smaller than a full case
- The spatial arrangement of the components is dictated by the frame design (you can't customize the footprint to work around space constraints)
- PC components aren't protected from loose objects in the cab

Considerations for using an open-frame case:

- Secure the frame in the cab so that it doesn't move around. Some open frames are specifically designed for lab-bench mounting, which would translate directly for a pin cab setup. Otherwise, you'll have to improvise something. I'd look for a way to fasten the floor of the frame to the floor of the cab with a few brackets or latches, so that it's not too much work to remove it.
- As with a conventional case, an open-frame case might conflict with the subwoofer placement in our WPC-style plans, so you should plan around that when building the cab.
- As with a conventional case, you'll have to tap into the frame's power button wiring, so that you can connect the motherboard's soft power switch to your cab power switch.

No case

Or, to put it another way, use the pin cab itself as the case. This is how I built my own cab, and I think it's the most common approach. I like it for its flexibility, especially the ability to position the components to make room for other things in the cab. But it's more work since you have to come up with custom mountings for everything.

Pros:

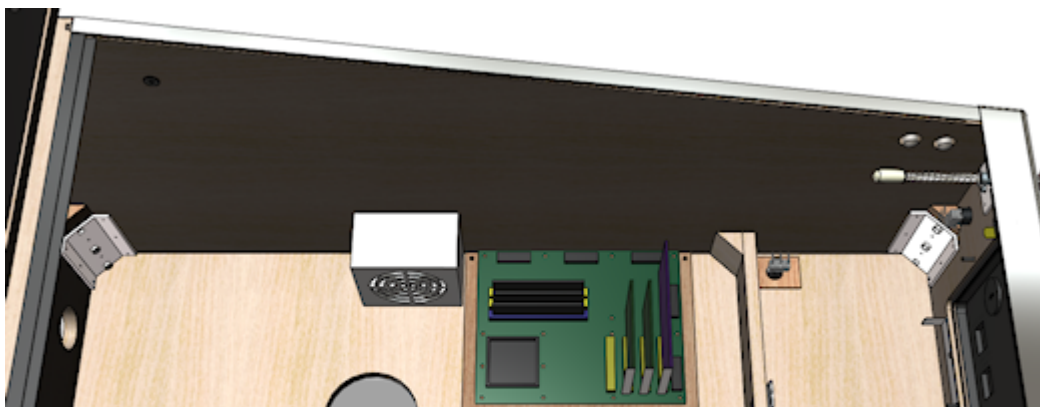
- Uses the least space
- You can arrange the components however you want, to minimize the footprint and work around conflicts
- Open air flow for cooling

Cons:

- More work to plan and implement
- Requires ad-hoc support apparatus for the video card
- No protection against loose objects
- More difficult to remove the PC components for service

Tips for a no-case installation:

- The motherboard can go anywhere there's room for it, but I'd recommend placing it in the middle of the cabinet front-to-back, and up against one side. You probably don't want to put it at the very front, because you need to leave room for the standard pinball equipment like the cashbox, tilt bob, plunger, and flipper buttons. You also don't want it at the very back, because that makes it hard to reach once everything else is installed.



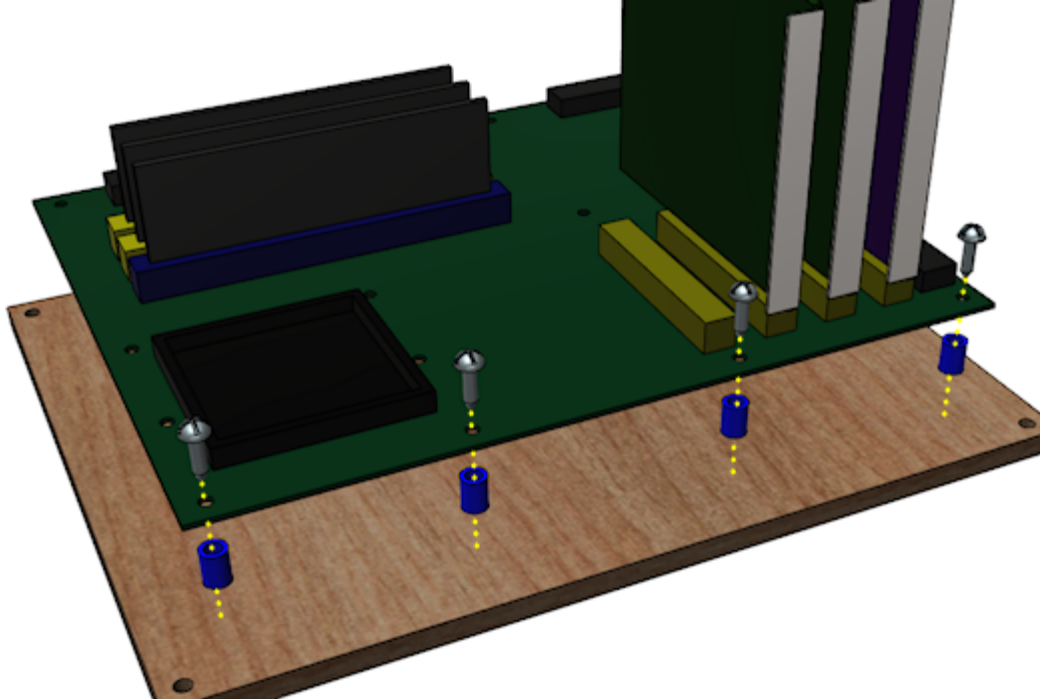


Possible placement of the PC motherboard and power supply. This leaves room along the side opposite the motherboard for external USB devices such as key encoders, plunger/nudge devices, and feedback device controllers.

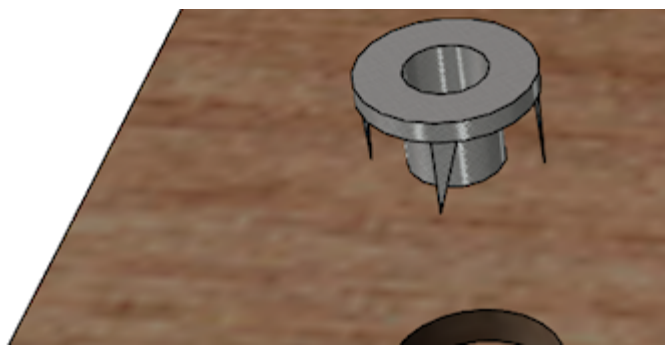
- The PC power supply can be placed up against the wall near the motherboard. The exact placement isn't critical; it just has to be within reach of the motherboard power connectors.
- The power supply (like the motherboard) needs to be fixed in place somehow. I secured mine with L-brackets screwed into the cabinet floor and wall. They're not screwed into the power supply itself; they just hold it place by forming a sort of cage around it.

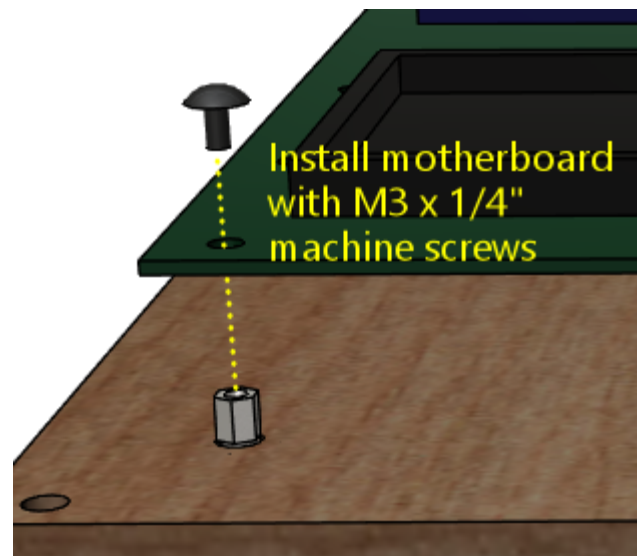
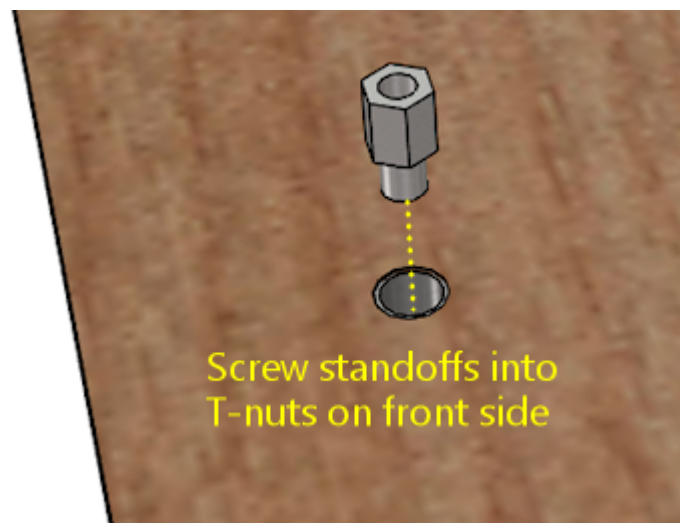
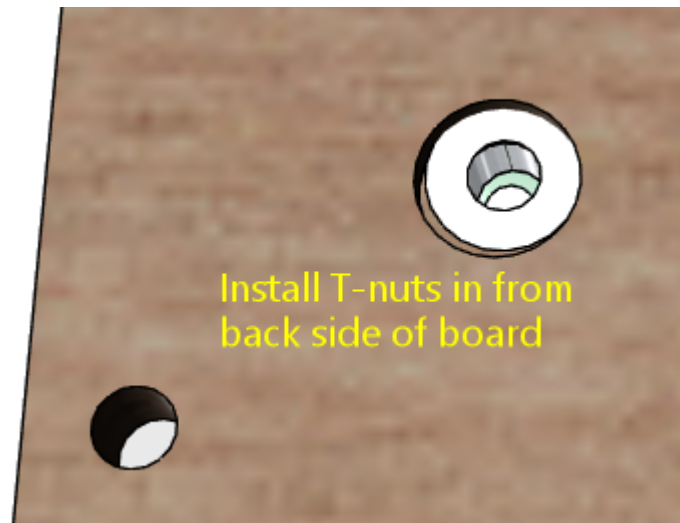
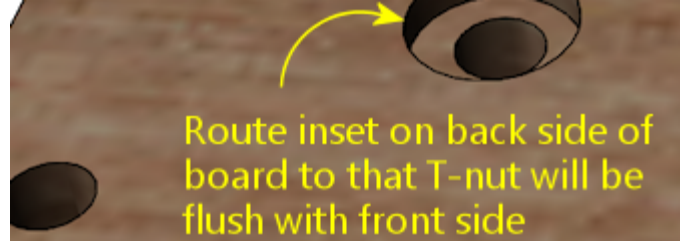


- I mounted my motherboard on a plywood carrier - so in a way it's just a home-brew version of the "open-frame" case, although with more control of the geometry than you get with a retail frame. This let me assemble the PC and its expansion cards outside of the cabinet, and drop the whole thing in as a unit. It likewise lets me remove the assembly when necessary to work on it outside of the cab. The plywood carrier is fastened to the cab floor with wood screws at the corners.
- PC motherboards attach to a conventional case with M3 machine screws that screw into standoffs, which in turn screw into threaded holes in the case. The standard standoffs won't work with a plywood carrier, though, because the plywood doesn't have the threaded holes that the standoffs fit into. As a replacement, you can use M3 or #4 by 1" sheet-metal screws, combined with 1/2" tall nylon spacers underneath. The sheet-metal screws will self-tap in plywood, so you can simply fit them through the spacers and screw them directly into the plywood. The spacers take the role of the standoffs to provide a little air space between the motherboard and the plywood carrier.



- Alternatively, it is possible to use the standard standoffs, but it requires some prep work. It's also much neater, so you might find it worth the extra effort. The trick is to pre-install a set of #6 T-nuts in the plywood to fit the #6 machine screw bases of the standard standoffs. T-nuts install from the *back* side of a board. You'll have to drill holes for the T-nuts, and possibly (depending on the thickness of the board) route insets on the bottom side so that the T-nuts seat flush with the front surface of the board. Once the T-nuts are in place, you can screw the standoffs into the carrier as though it were a regular case chassis, and then install the motherboard on top of the standoffs using M3 x 1/4" machine screws.

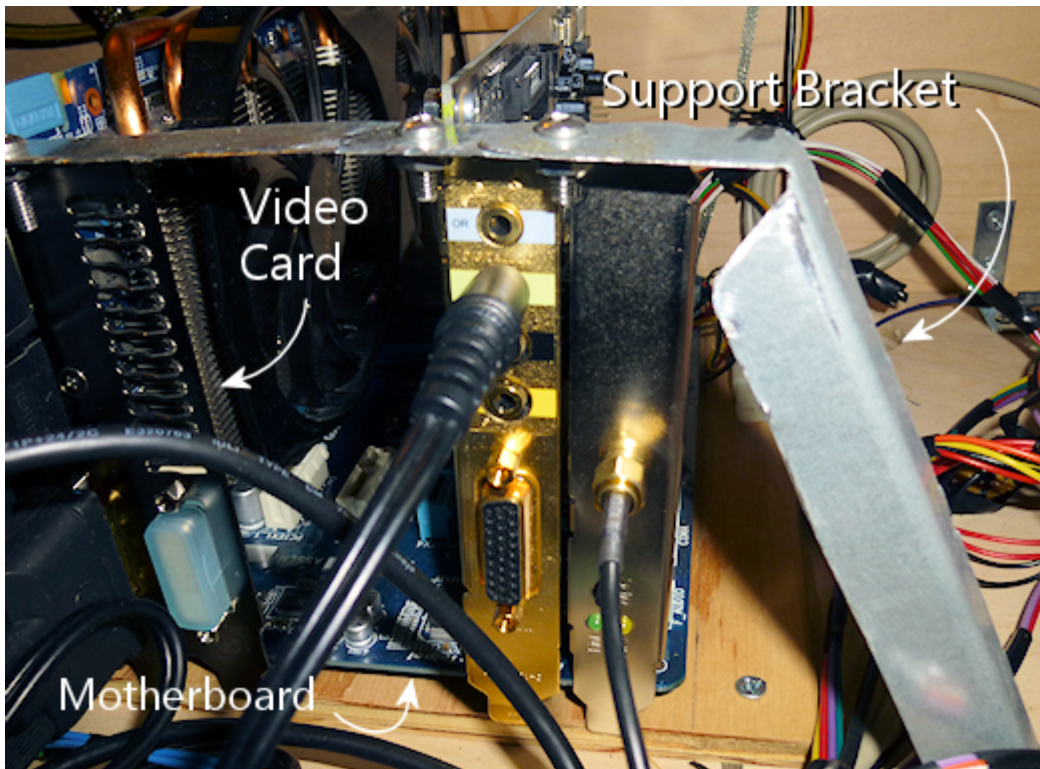




- Your video card and any other expansion cards will need a support strut of

some kind to keep them from moving or coming loose. The slot on the motherboard isn't strong enough to serve as the only physical support. Without more support, the cards can wiggle in their slots, and that can momentarily disconnect their pins in the PCI slot. That's very bad, because PCI slots most definitely aren't "hot pluggable" - they don't let you insert or remove cards with the power on. At best, a momentary disconnect will crash Windows and make the PC do a hard reset; at worst it could damage the video card or motherboard. So the video card and other cards must be held rigidly in place.

For my setup, I made a metal bracket that runs between the plywood carrier and the top of the video card at a 45° angle. It attaches to the video card via the same screw hole that you'd use to screw the card into a regular PC case frame.



- Hard disks can be placed anywhere convenient that's within reach of the data cables and power connectors. If you're using an SSD (solid-state disk), those are so light that you can secure them to the floor or a wall with adhesive Velcro. I'd probably even use Velcro for a hard disk, although due to the extra weight, I'd only mount that flat on the floor rather than trying to stick it to a wall. That way, gravity will be working with the Velcro rather than against it. I've had problems with Velcro on a vertical surface sliding down the wall over time, so I wouldn't trust it to hold a mechanical hard disk up against gravity. If you want to mount a hard disk vertically, use something more robust to secure it, like L-brackets with screws.
- This is more of a PC-planning thing, but you can avoid the need for a separate disk of any kind by using an M-SATA drive. Those are SSD/flash drives that are only about the size of a credit card and plug directly into a slot on the motherboard, so there are no cables to manage and no separate piece to mount. You need a motherboard with an M-SATA slot to make this work, obviously.

External case

A rather different option from what we've been talking about thus far is to "think

outside the box": placing the PC in a conventional tower case sitting on the floor next to the cab.

This doesn't yield the kind of integrated, self-contained, stand-alone pinball machine replica that most pin cab builders are shooting for, so it's definitely not to most people's taste. But it has two features that might make it interesting to some people. The first is that it lets you share a PC between the cab and other uses, such as using it as your regular desktop PC most of the time. It's hard to deny that buying a whole separate PC that will never be used for anything but playing pinball is a bit of an extravagance. By the same token, it's hard to deny that building a giant piece of furniture just to play video pinball is an extravagance; but sharing the PC is one way to rein in the cost where you can. The second thing that might make an external PC attractive for some projects is that it uses zero space inside the cab. That's not much of a concern for a full-size cab, but if you're building a sufficiently "mini" mini-cab, you might not have all that much free space to work with - or you might want to save the space for something else, like more feedback devices.

Pros:

- Saves space inside the cab
- Lets you use the PC for other functions when not pinballing

Cons:

- Not self-contained
- You have to connect several cables each time you use it

The only integration work with an external PC is to run the video and USB cables between the PC and the cabinet. You'll probably want to minimize the number of cables, both to reduce visual clutter and to make it easier to connect and disconnect cables when you want to switch the PC between pinball mode and desktop mode.

My first suggestion is to place a USB hub inside the cabinet. Connect all of the USB devices in the cab to the hub; all of that wiring will be hidden inside the cab, and you won't need to touch any of it when you connect and disconnect the PC. Then you just need one external USB cable, between the PC and the hub.

My second suggestion is to use Keystone or similar wall plate connectors for the USB cable and all of the video cables. See "External I/O plugs" below for more on this. I'd set up all of the external ports on the back wall of the cab. That will at least centralize all of the cable connections in one place, which will make it easier to keep the cable bundle neat, and will also make it easier to plug and unplug everything when you move the PC. Putting the connectors in the back will keep the rat's nest of cabling as out of view as possible.

External I/O plugs

If you're using a wired keyboard or mouse, or a wired Ethernet connection, you'll need a way to plug these into the motherboard from outside the cabinet.

A simple way to accomplish this is to drill a hole in the cabinet big enough to accommodate the wires, then simply pull the cables through the hole and plug them into the motherboard. This isn't very convenient when you need to unplug or reconnect the cables, though, since you'll have to open up the cabinet to get access to the plugs.

A better way is to install a set of the appropriate jacks on the exterior of the cabinet. Then you can easily plug the devices into the external jacks at any time, and just as

easily unplug them. This is fairly easy to set up, thanks to connectors you can buy for home theater and office installations, where many people want to put USB connectors and Ethernet cables in wall outlet plates.

For the keyboard and mouse connector, I recommend placing two USB connectors (one for each device) on the floor of the cabinet near the front. For the Ethernet connection, I recommend putting a port on the back wall of the cabinet near the power inlet.

Warning! Before you install these jacks, you should figure out how all of the internal parts in your cabinet are going to be laid out, so that you're sure the chosen locations for the jacks won't get in the way of anything else. The jacks require cutting holes in the cabinet, so you can't easily move them if the locations end up conflicting with something else.

USB, keyboard, mouse

For the keyboard and mouse ports, here are the parts I recommend:

- Keystone wall plate insert with 2 openings
- Keystone snap-in USB 3.0 female-to-female couplers (quantity 2)
- Keystone PS2 (6-pin mini DIN) female-to-female coupler (optional, if you're using an older keyboard with a PS2 connector instead of USB)
- 4-foot standard USB cables (male A to male A) (quantity 2)
- 4-foot PS2 keyboard male-to-male cable (optional, if you're using the PS2 keyboard connector instead of USB)

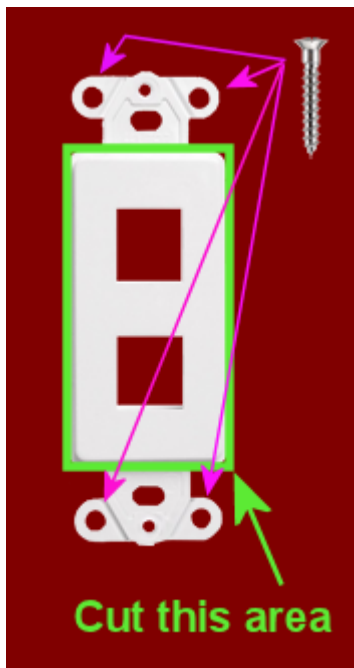
Note: you can get a 3-gang plate if you want to install the keyboard, mouse, and ethernet in one spot, and you even get a 4- or 6-gang plate if you want to add even more connections beyond these. I personally prefer having the keyboard and mouse connections near the front of the machine, since that's where I tend to use those devices, and I prefer to have the Ethernet port in back to keep that cable out of the way. That's why I recommend separate plates for these - it's simply because I like to locate the two sets of plugs at opposite ends of the machine. But if you want to minimize the number of holes to drill, you might prefer to put everything in one plate instead.

To install: figure out where the plugs will go. I recommend the floor of the cabinet near the front. Using the wall plate insert as a template, mark the area on the floor inside the cabinet where the rectangular middle section of the plate will go. This is the area to cut out. Note that you should mark this area from the inside to make sure there's room for the screw tabs on either end of the plate.

If you're using a jigsaw to cut this out, drill a hole at one corner big enough to start the cut, then carefully cut the rectangular area out with the jigsaw. If you're using a router, carefully route along the cutting outline.

Once the hole is cut, mount the plate on the inside of the cabinet, with the outside of the plate facing out through the opening. Screw down the plate with four 1/2" wood screws, one through each screw tab.

Insert the two USB couplers (or one USB coupler and one PS2 coupler, if you're using that type) into the square openings from the inside of the cabinet. They should attach so that the jack on the underside (exterior) of the cabinet is flush; the body of the coupler should stick up into the interior of the cabinet. Grab the matching 4' cables and plug one end of each cable into the corresponding coupler. Plug the other end of each cable into a suitable port on the motherboard.



That's it! This will give you a neat, recessed opening with the two jacks for the keyboard and mouse. You can now plug your devices into the jacks without having to open up the cabinet.

Ethernet

The procedure for the Ethernet connection is the same as for USB ports. Here are the required parts:

- Keystone wall plate insert with 1 opening
- Keystone Cat6 (RJ45) female-to-female coupler
- 4' standard Ethernet Cat6 patch cable

As before, find the location in the cabinet where you'd like to install the port. I recommend putting this connector on the back wall of the cabinet near the power inlet, since it's tidier to keep the Ethernet cable behind the machine rather than having to route it underneath. Repeat the procedure to mark and cut the opening required for the plate. Install and screw in the

plate as before, pop in the Ethernet coupler, and connect the short Ethernet cable between the coupler and the motherboard Ethernet jack. Now you can plug an external Ethernet cable between the external cabinet jack and your router or wall plate.

HDMI

Keystone HDMI coupler modules are available that work just like the USB and Ethernet modules. Follow the same procedures as described above.

Other video

DVI-D, DisplayPort, and VGA connectors are too large for the standard Keystone jacks, so you generally can't find these as Keystone modules. However, you *can* find non-Keystone wall plates specially made for the various video connectors. So if you're using an external PC and need a set of video connectors, don't give up when you can't find them in Keystone modules; instead look for one-off dedicated plates for the video connector types you need. The procedure to set those up is generally the same as for the Keystone plates.

28. Cooling Fans

The TV and the PC components inside a pin cab generate heat when running, so it's necessary to ventilate the cabinet to keep it cool. Most pin cab builders accomplish this by cutting a couple of air vents in inconspicuous places on the cab and installing fans in the vents to move air through the cabinet.

Most cab builders use PC case fans for this. They're a good fit because they're designed for exactly this type of use, and they're quiet and inexpensive. They also fit well with our electrical setup, since they're built to work with a standard PC power supply, which we'll already have to run the PC components.

How many fans?

It seems pretty standard to use two exhaust fans at the back of the main cabinet, using common PC case fans. Most people also add one or two intake fans on the floor of the cabinet. Passive intake openings (without additional fans) will also work, since the pressure from the exhaust fans will draw air in through the intakes anyway, but intake fans will help increase the air flow.

This design is driven more by the practicalities of available space than anything else. The back wall of the cabinet can easily fit a couple of PC case fans, and you can usually find room on the floor of the cabinet for at least one more, as an intake. I haven't attempted any sort of quantitative analysis of how much cooling is needed, or how much cooling this fan arrangement provides, but I think it's pretty well proven in practice. Lots of people have built cabs with roughly this setup, and I don't think I've ever seen reports on the forums of significant problems with heating.

Fan placement

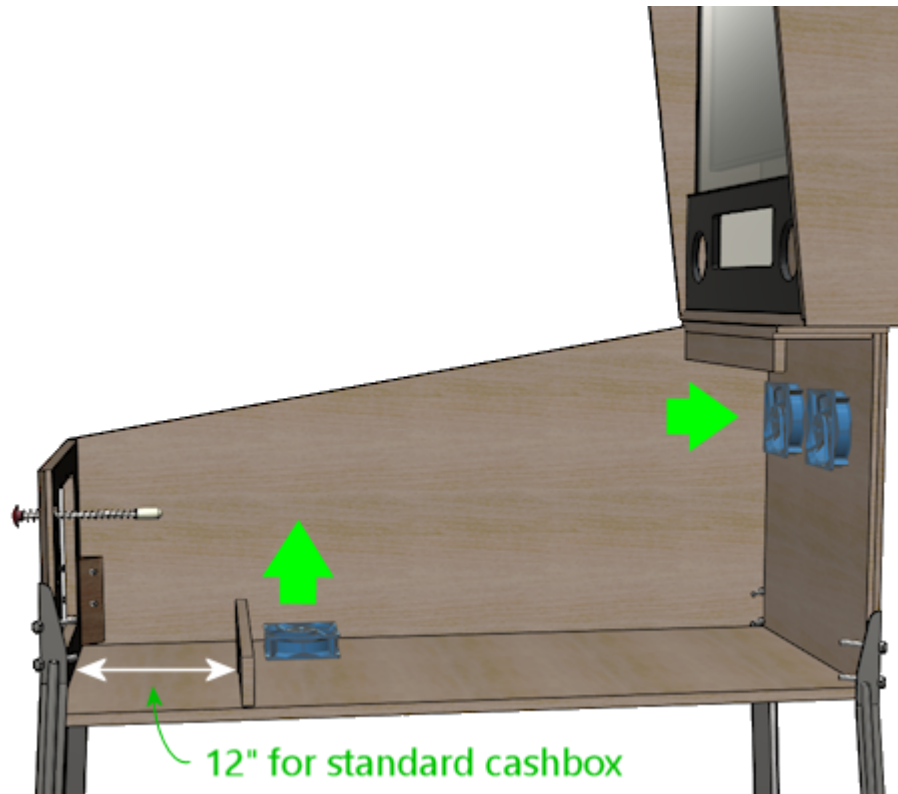
Our goal with the ventilation system is keep air moving throughout the entire cabinet. We don't want any stagnant pockets of air that can sit there and gather heat; we want to continually discharge the hot air within the cab and replace it with fresh air from outside.

The best way to accomplish this is to put vent openings at each end of the cabinet, front and back, and drive air through the vents with fans at each opening. The fans should be oriented so that the fan at one end is drawing air into the cabinet, and the fan at the other end is blowing air out of the cabinet.

One set of fans should be installed in the back wall of the cabinet. This is an obvious place in terms of cosmetics, since the back of the machine isn't normally visible. It's also good in terms of air flow, since it's at one end of the cabinet in the long direction. What's more, you can take advantage of the tilt of the playfield TV. The TV is normally installed at an angle, rising towards the back. Since hot air naturally rises, heated air will tend to move towards the raised back end of the playfield. Take advantage of this natural convective flow by placing the rear vent(s) fairly high up on the back wall, and orienting the fans so that they blow air *out* of the cabinet.

The other set of fans should be installed near the front of the cabinet. You almost certainly won't want to cut fan vents in the front wall, for cosmetic reasons, so most people place the front fan or fans in the floor of the cabinet, close to the front of the cab. To maximize air flow, orient the front fan(s) to draw air into the cabinet. The positive pressure (into the cabinet) of the front fans will combine with the negative pressure (out of the cabinet) of the rear fans to maintain a constant flow of air front-to-back.

If you're planning to use coins with the coin chutes, remember to leave room at the front of the machine for a container to collect inserted coins. There's a standard coin tray design used on the real machines, described in the "Cashbox" section in Chapter 23, Cabinet Hardware Installation. That requires about 12" of clearance at the front.



Typical cabinet fan arrangement. One or two fans are placed in the floor near the front of the cabinet, with the blades oriented to draw air into the cabinet. Leave room for the coin collector box at the front (12" for a standard pinball cashbox). Another set of fans is installed high up on the back wall, oriented to blow air out of the cabinet. This will maintain air flow through the whole cabinet from front to back.

Selecting fans

Most cab builders use 120mm PC case fans, similar to the type pictured at right. The "120mm" refers to the diameter of the opening; 120mm is about 4¾ inches.

Case fans come in several sizes, from 70mm to upwards of 200mm. Larger diameter fans are generally quieter, and gamers building high-end rigs can sometimes get so obsessed about it that fan size can seem like a fetish. If you could fit a ceiling fan into a gaming rig, that would probably be a thing. In my experience, though, you can get good results in a pin cab with ordinary 120mm fans. Good ones can be pretty nearly silent, certainly quiet enough to be inaudible over the pinball action.



Like most PC components, case fans have a standardized form factor that allows any fan of a given standard size to fit any case designed for that size. The fan housing includes screw holes at the four corners for attaching it to the PC case. For our purposes, we can use wood screws through these holes to attach the fan to the cabinet wall or floor.

I'd recommend looking for a fan with a 3-or 4-pin Molex plug like those pictured below. This is the most common design in current use, so it's the easiest to find and the most likely to be compatible with your motherboard. These plugs are designed to fit into a mating connector on your motherboard. Note that the 3-pin plugs are compatible with 4-pin motherboard connectors (no adapter needed), so you can buy a fan with either type if your motherboard has 4-pin connectors.

If your motherboard doesn't have a fan connector at all, you'll have to plug your fans directly into the ATX power supply. For a fan with a 3- or 4-pin Molex fan plugs, you can do this with an adapter cable.



3- and 4-pin fan plugs, for connecting to your motherboard's "SYS FAN" or "SYSTEM FAN" headers.

I don't have any particular brand recommendations. Fans are standardized enough to be interchangeable, so once you decided on a size, shop by price and/or user reviews.

Cutting the openings

Once you decide where to mount your fans, you should simply cut circular openings in the cabinet wall and/or floor of the appropriate size. If you're using 120mm fans, cut a circular opening 120mm in diameter for each fan.

You don't have to get the size of the opening exact, since the fan itself doesn't have to fit into the hole. The opening is purely for the air vent. The fan itself will simply mount flush against the inside surface of the cabinet wall or floor, covering the opening.

Installing the fans

Before installing the fans, figure out which way it blows air so that you can install it with the air flowing in the right direction, into or out of the cabinet.

How do you tell the direction the fan goes? Look for an arrow printed or stamped on the side of the housing: if you can find one, it should indicate the direction of air flow. Failing that, you can simply connect the fan to power and check which way the air is blowing.

Once you determine which direction the fan goes, simply position it over the opening with the appropriate side facing outwards, according to whether you want it to serve as an intake fan or an exhaust fan. Attach it to the cabinet wall or floor with four wood screws. #4 wood screws should work well for most fans.

Powering the fans

The next step is to connect the fans to power.

If your fan has a 3-pin or 4-pin Molex connector (like the ones at right), look for a SYSTEM FAN or SYS FAN connector on your motherboard. Most modern

motherboards have one or two of these connectors.



Motherboard fan connector. These are usually labeled SYS FAN or SYSTEM FAN.

Most modern motherboards have at least one fan connector, usually two.

You can safely plug a 3-pin fan plug into a 4-pin motherboard fan header. The connectors have keying slots to make sure the pins line up correctly even if mixing 3- and 4-pin connectors. As long as the fan plug physically fits the header, you can just plug it in directly without any sort of adapter.

If you have more fans than available headers on your motherboard, you can buy a splitter (also known as a "Y" cable) that lets you connect multiple fans to one motherboard header.

If your motherboard doesn't have any fan connectors (which is uncommon), you'll have to plug the fan directly into the power supply instead. ATX power supplies don't typically include connectors for the small fan plugs, though, so you'll need an adapter cable. You can find these at PC component retailers: search for "fan ATX adapter cable", and look for a cable with a male fan connector.

If your fan has a larger 4-pin connector like the one pictured at right, it's designed to plug directly into your ATX power supply rather than plugging into the motherboard. You should find several mating connectors on the power cords coming out of your power supply. You can simply plug the fan connector into the matching power supply connector.



Extension cables

You'll probably need a longer cable than what's attached to the fan. The fan's built-in cable will be designed for the relatively confined area of a normal PC case. Full-size pin cabs are quite a lot larger, so the fans will probably be further away from the motherboard than in a regular PC.

You can buy a fan extension cable from a PC parts vendor if necessary. Alternatively, if you don't mind doing some soldering, you can simply cut the existing fan wires in half and solder as much additional wire as you need between the two segments to extend the cable length. If you do this, be sure to wrap the exposed solder joints with electrician's tape to insulate the wires.

Backbox cooling

Some cab builders also put a fan or two in the backbox, to provide active ventilation for the TV there. Others use passive ventilation - no fans, just vent holes in the rear wall of the backbox.

If you're using an older TV with a display technology that generates significant heat, such as a plasma TV or an LCD TV with a fluorescent backlight, a fan is worthwhile. Newer LCD panels with LED backlights run cool enough that a fan probably isn't

necessary, as long as you provide good passive ventilation.

The standard cabinet design for most real machines in the 1980s and 90s used passive ventilation, typically with seven 1½" diameter holes running across the width of the back wall, located about 1" from the top and spaced 1" apart.



Typical backbox passive ventilation holes used in real pinballs from the 1980s and 90s. Seven holes are drilled across the width of the backbox's rear wall. Each hole is 1½" in diameter; holes are spaced 1" apart and 1" from the top.

Note that the standard backbox design allows for some air movement between the main cabinet and the backbox, via the large opening in the floor of the backbox. That's why the ventilation holes are only needed at the top of the backbox: as warm air rises through the backbox and exits via the top vent holes, cooler air will be drawn in through the cabinet opening. If you don't use the standard design with the opening between the backbox and main cabinet, you should add some air intake holes at the bottom of the backbox.

Is passive ventilation really enough for a TV? Let's consider how much heat the traditional design was intended to handle in a real machine, and compare that to our needs for a virtual cab. We'll use electrical power as a proxy for heat. The real machines housed their main control electronics in the backbox, along with their score displays and about a dozen small incandescent bulbs for lighting the backglass artwork. The total power usage of all of this equipment adds up to about 50W. A 32" LED-backlit TV runs at about 55W. TVs will probably get more efficient as time goes on, plus a backbox TV is usually a little smaller than that (which usually translates to less heat), so that 55W estimate is probably erring on the cautious side.

In other words, our TV should produce a pretty similar amount of heat to what was in a real machine, so if the passive cooling was good enough for the real thing, it should be adequate for a virtual cab as well.

On the other hand, it seems that we don't have a lot of headroom here: a TV will use up most of our estimated heat budget. If you're also installing other backbox elements that generate significant heat - particularly a plasma DMD - active cooling might start looking like a good idea.

If you do decide to include a fan in the backbox, I can suggest two configurations:

- The first is to keep the passive vent holes near the top, in the same arrangement described above, and add the fan as an intake near the bottom of the backbox. The positive pressure of the fan will combine with the natural

chimney effect of rising warm air to maintain steady air flow.

- The second is to remove the passive vent holes and replace them with one or two openings for, say, 120mm PC case fans, also near the top of the machine. Install these PC fans in exhaust mode, with the fans oriented to blow air outwards. This will draw in air from the main cabinet and blow hot air out the top.

Be sure to consider space conflicts between the fans and the TV and other backbox elements.

CPU fans

The CPU chip on your motherboard will probably require a separate fan mounted directly on top of the chip. The CPU generates a great deal of heat in a very concentrated area, so this fan is needed to quickly move heat away from the surface of the chip.

The CPU fan shouldn't require any extra cabinet planning or cutting, since it mounts directly on top of the CPU chip itself. It's basically part of the CPU/motherboard assembly. If you buy your CPU in retail packaging, a matching fan is usually included, so the only thing you have to do is install it when you assemble the motherboard. If you buy an unpackaged "OEM" CPU, you'll probably need to buy a fan separately. You have to buy a fan designed for the particular chip type, because it has to match the CPU's size and shape. You should be able to find suitable fans on Newegg.com or other sites that specialize in PC components.

29. Playfield TV Mounting

This section is about how to install the main TV - the one that takes the place of the playfield.

This section tries to address the two big questions about installing the TV. The first is where to position the TV in the cabinet, which is as much a matter of aesthetics as of function. The second is the engineering question of how to physically attach the TV to the cabinet, once you've figured out where you want it.

Where to place the TV is one of the top questions that new cab builders ask on the forums. It's one of those things that seems self-evident at first glance, but has a lot of subtlety when you look more closely. It's obvious that the TV has to lie on its back near the top of the cab, but that doesn't quite answer the question, since it leaves a few inches to play with to move the TV up and down and front to back. Those few inches can make a lot of difference aesthetically - it's that matter of what *exact* positioning is ideal that raises all of the forum questions. This chapter surveys the options and their respective pros and cons. I have some opinions about what looks best, which I'll share, but I'll also try to give equal time to the alternatives. I don't think there's a single right answer, because everyone has their own priorities for their cab and their own sense of what looks best.

After the survey of positioning alternatives, this section presents my attempt at an all-purpose, universal TV mounting system. When I was building my own cab, I found the TV mounting to be one of the more challenging problems. The TV makers don't expect you to use a TV like this, and the people who make pinball machines don't think of putting a TV inside, so neither world gives us an example we can look to for ideas, and neither world offers a ready-made hardware solution we can apply. Every cab builder has always been on their own to work out their own unique, ad hoc scheme. That always seemed like a waste of energy to me; I've always thought there should be at least a basic template we can follow. I think I've managed to come up with something like that. This chapter provides a general-purpose design that should be flexible enough to work with most TVs and most cabs, using standard parts that you can readily buy. It's at least an option to consider, and even if it isn't a good fit for your cab, it might still give you some ideas to draw on. If you're looking for even more ideas, at the end of the chapter, I'll outline a few alternative mounting schemes that other cab builders have used.

We're assuming here that you've already picked out a TV to install. If you're still shopping for a TV, there's a separate section with advice about that, Chapter 7, Selecting a Playfield TV.

Orientation

The playfield TV is always installed in "portrait" mode, to fit the proportions of the cabinet. This represents a 90° rotation from the standard way you view a widescreen TV.

But should it be 90° clockwise or 90° counter-clockwise?

Most virtual cab builders install it with the **bottom of the TV** facing **left** in the cab:





In principle, it really shouldn't matter. Windows and all of the pinball software *should* let you select whatever rotation you want. But this is one of those cases where you can save yourself some hassle by doing it the same way everyone else does it. There's still a lot of older software in use in the pinball ecosystem, and some of it might not be as adaptable as modern programs. I don't know of any actual examples of software that will outright fail to work with other rotations, but I know from helping people out with PinballY that monitor rotation in general can cause configuration headaches, especially with some of the commercial games.

Positioning options

Before we get to the mechanics of installing the TV, let's consider exactly where you want to position it. To a first approximation, of course, it goes "where the playfield goes". But a TV isn't quite the same as a regular playfield, in either its nature or its size and shape. So saying that the TV goes where the playfield goes is too vague to constitute a plan. It leaves a couple of important questions unanswered:

- Should the TV screen be flush with the top glass, or set in a bit like a real pinball machine's playfield?
- Should the TV be all the way at the front, flush with the lockbar, or should you set it back a few inches to make room for a plunger?

Judging by how often these questions come up on the forums, many new cab builders agonize over these quite a bit.

I want to offer some thoughts about how to decide these questions. Most new cab builders focus on how the placement will affect *playability*. That's certainly the right priority. But the thing is that playability actually won't be much affected, no matter what TV positioning you go with. When you're playing, your eye adapts to whatever setup you have, and before long you won't even notice it. It's the same principle that makes the curtains around a movie theater screen disappear once the film starts rolling. When you're playing, it doesn't much matter where the TV is relative to the rest of the cab. All of that disappears; your eye pays attention to the table. This should be reassuring if you're been agonizing over the question, because it means that you'll likely be happy with your cab's playability no matter what you decide.

TV placement can make a big difference to the aesthetics of the cab, though. Since

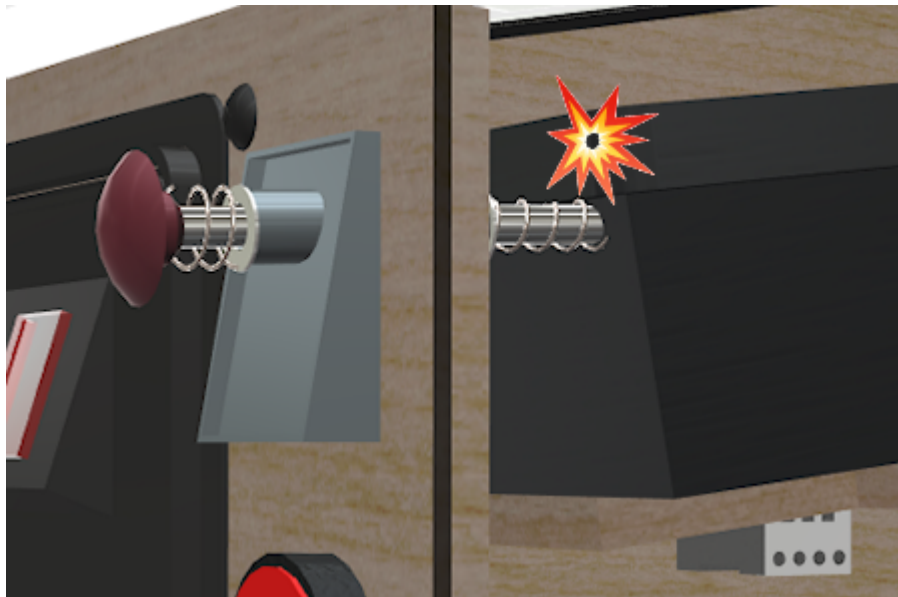
playability isn't much of an issue, I think the aesthetics are the better focus when making these decisions. In addition, there are some functional considerations, since the TV placement affects the space layout inside the cabinet. You might have space constraints that decide some of these variables for you, before you even get to think about how it'll look.

The dreaded plunger space conflict

See also "Positioning the plunger" in Chapter 37, Plunger.

One of the key space constraints that affects TV placement is the plunger. This is an issue because the most natural placement of the TV and of the plunger put them into conflict: they both want to occupy the same space.

The natural place for the TV is all the way at the front of the cabinet. The natural place for the plunger is where it goes on the real machines. The problem is that the plunger sticks into the cabinet by about 3", so if the TV is all the way at the front, it overlaps the plunger.



Why isn't this a problem on the real machines? In part, it's because the plunger sits just above the playfield on a real machine, so they're in different planes vertically and thus don't collide. In addition, on the real playfields, they cut a notch out of the playfield at that corner specifically to make room for the plunger. That lets you lift up the playfield without hitting the plunger.

If we could cut a notch out of the TV, we could solve this the same way, but that's not really an option. Our options all involve moving either the TV or the plunger to make room for the other:

- Move the TV down a couple of inches to clear the plunger. I don't think I've ever heard of anyone doing this; I think it would put the TV lower than anyone wants it. In addition, you'd now have the problem that the plunger sticks out visibly above the TV, so you'd have to do something to cover that up.
- Move the plunger upwards far enough to clear the TV. I don't think this would be practical given the space constraints, but maybe you could make it work by sharing the work: move the TV down slightly and move the plunger up slightly. The plunger would stick out above the TV, so you'd have to cover it up somehow.
- Get rid of the plunger and use a Launch Ball button instead. The button doesn't

create the same space conflict, so you can put it where the plunger would normally go and still have the TV all the way at the front. If you're not particularly attached to the idea of a plunger, this option has the additional upside that it's a big simplification overall, in that plungers are complicated on a virtual cab by their very nature. A lot of us would never consider doing without a plunger, though, since it's such an iconic pinball element.

- Move the plunger down to clear the TV. This requires moving it down by about 3". Some cab builders take this approach because it lets them both have a plunger *and* put the TV exactly where they want. I personally don't like the resulting look, though - it gives it a kind of weird home-brew vibe and makes it too obvious that it's not a real pinball machine.
- Move the TV further back to clear the plunger. This has the downside that it creates a gap at the front of the cab before the TV starts, which is unacceptable to some cab builders (who want the TV at the very front). In my opinion, though, a gap at the front isn't a problem, and you can even turn it into a virtue. For one thing, you can fill the space with something resembling the apron on a real machine. I think a 3D element like this adds some nice texture to the otherwise flat expanse of TV screen, and it's an opportunity for some added decorative graphics. For another, you're going to have a front-to-back gap *somewhere*, because a 16:9 TV simply doesn't have the same proportions as a standard cab. If you put the TV at the very front, the gap all ends up at the very back. I think it creates a more balanced look to split the gap between the front and the back.

Which option is best comes down to the priority ranking you would assign to these three goals:

- I want a plunger
- I want the launcher at normal height
- I want the TV at the very front

Basically, you get to pick two of these, but you can't have all three. Pick the two that you think are the most important, and that decides the question for you:

- I want a plunger, I want it at normal height: then you should move the TV back. Personally, I rank these priorities highest, and this is the solution I like best.
- I want a plunger, and I want the TV at the very front: then you have to lower the plunger to clear the TV. I think that looks weird, but tastes vary.
- I want the launcher at normal height, and I want the TV at the front: then you have to dump the plunger and go with a Launch Ball button. I wouldn't want to forego the plunger, but not everyone feels as strongly.

Inset depth

The first decision you have to make about TV positioning is whether to install the TV screen flush with the top of the cabinet, or recessed into the cab by some distance. In the latter case, most cab builders think of this in terms of placing the TV at the normal playfield depth of the real machines.

These two main options are illustrated below, for the sake of clarifying our descriptions, but I wouldn't try to make an aesthetic judgment from the diagrams alone. The differences in question are subtle enough that it's hard for an illustration or photo to capture the full effect.

First, placing the TV flush with the top:



Playfield TV flush with the top of the cabinet, taking the place of the top glass. The top glass can be added if you set the TV back by about 1/4" to make room.

And second, recessing the TV into the cabinet to about the depth of a normal playfield:



TV at roughly the same depth as a normal pinball playfield.

In comparing these for aesthetics, note that we've made the "filler" areas at the top and bottom more conspicuous than they'd be in a real build. You'll probably make these a darker color in your actual build (probably black, maybe with some graphics decorations). We wanted to make it obvious in the illustrations that they're not part of the TV screen, which they might appear to be if we made them a flat black.

Pros and cons

Aesthetically, I have a strong preference for the inset version. I've seen both setups in person, and I find the flush-top version to look too much like a video game, with the whole top being a TV screen. Setting the TV screen down into the cabinet makes it look more like a regular pinball machine, and creates more of a 3D effect. It also lets you add a raised apron at the front, which adds another 3D element to contrast with the flatness of the TV screen.

Some people prefer the flush-top version on the theory that the simulated pinball tables *already* depict a portion of the inset depth. I don't find that reasoning convincing, because most of the pinball programs let you adjust the point of view to show as much or as little of the side walls on the TV as you want. The less the better, as far as I'm concerned, because video images of the walls take away TV space that could be used for actual playfield area, and they don't look as realistic as real side walls.

In terms of playability, I don't think it makes any difference one way or the other. For the most part, once you're into a game, your eye only pays attention to the active playfield, and mostly ignores the surroundings.

Functionally, each version has its advantages. The inset version makes room for a flasher panel at the back, which I see as a major plus, as well as LED strips along the sides. It also leaves an air gap for cooling between the screen and the top glass (if you're including top glass).

The flush-top version has the advantage that it rotates the screen slightly closer to a head-on viewing angle. Everyone knows that the picture degrades on many flat-screen TVs when viewed from too steep an angle off to the side, so this might be a concern for some TVs. However, I think a lot of cab builders get overly worried about this. Keep in mind that the viewing angle difference between the "inset" setup and the flush-top setup is only about 2-3°, and they're both pretty far from head-on. I think on most TVs it will make little or no difference. If you're concerned about it, test your TV from the two angles and see if it makes a big enough difference to be the deciding factor.

A second advantage of the flush-top setup is that it consumes a little less vertical space in the cabinet. That's usually not a big deal one way or the other in a full-sized cab, but the extra space might make a bit difference in a mini-cab.

What is the standard real playfield depth?

If you're using an inset to simulate the playfield depth of the real machines, what's the authentic distance?

We actually have a fairly large range to choose from, because the real machines vary quite a bit, mostly by vintage. In the 1970s and earlier, most machines had very shallow playfield insets: the playfield surface was typically only about 1½" to 2" below the top glass, all the way from front to back, and the top of the apron was almost flush with the glass. In the 1980s, the depth started to increase to make room for the three-dimensional features that became common, such as ramps and two-level playfields. A mid-1980s machine might have an inset of 3" at the front and 6" at the back. Note that this generation started sloping the playfield relative to the

top glass, so that it had more headroom at the back, to allow for taller ramps and other features. As the years went on, the 3D features got even taller. By the 1990s, the playfield depth had increased to around 4" at the front and 8½" at the back. It reached a plateau at that point; the latest Stern machines tend to be about the same.

Recommendations

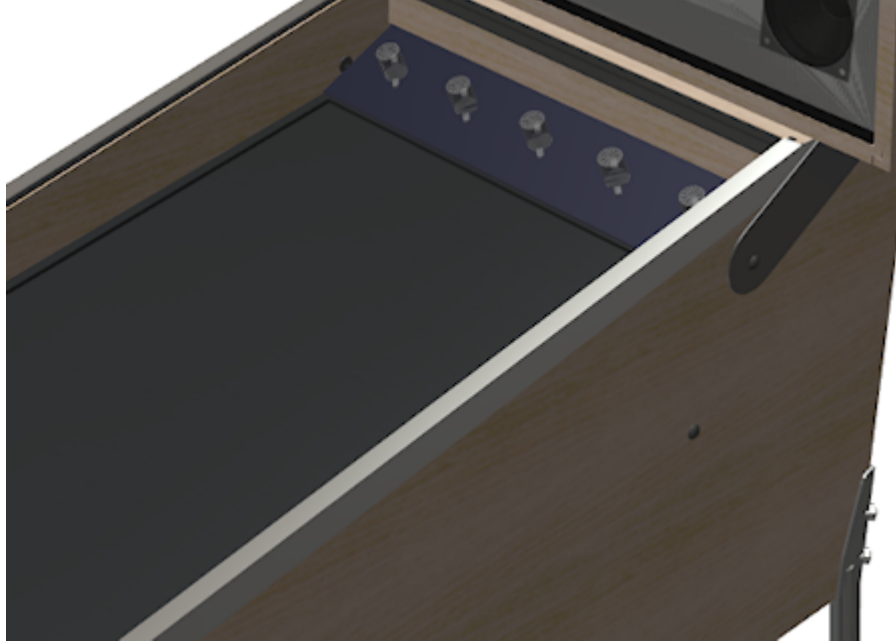
I much prefer the "inset" style over the flush-top design aesthetically, so that's my first recommendation. I'd only use a flush-top design if you have to due to some kind of physical constraint, like an oversized TV that can only sit on top of the side walls.

Assuming you're going with the inset style, the depth and angle are pretty flexible, since the real machines cover such a wide range. I don't think you need to replicate the precise measurements of any particular real machine - I think all you need to do to look "right" is to maintain the general proportions. Specifically, I'd say this means:

- the playfield should be set in by at least the height of the ball (about an inch)
- it should be angled slightly upward (about 5° to 6° relative to the floor)
- for WPC-style cabinets, the angle should be less than the angle of the top glass, so that the back of the TV is set in deeper than the front
- for older EM-style cabinets, the angle should usually be about the same as the top glass

In terms of looks, that gives us a pretty wide range to work with. There is one practical consideration that I'd add: if you're using an apron at the front and/or a flasher panel at the back, you'll need to leave a little extra vertical space for those. Exactly how much depends on how you want those features to look. For example, the flasher panel can be horizontal, tilted, or vertical:



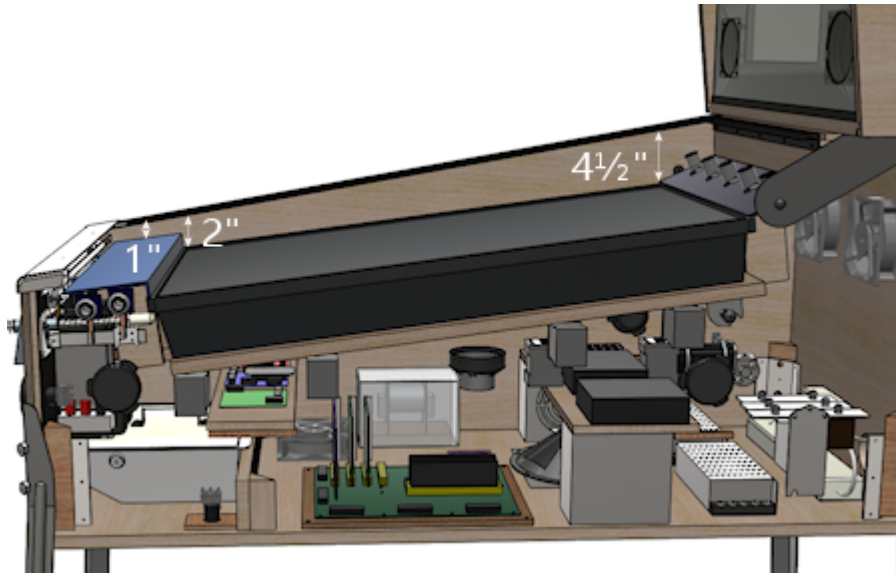


I personally like the tilted style best, but that's probably a matter of taste more than anything else. In terms of space, a flat flash panel only needs about 1¼" of headroom, for the height of the domes. A tilted flasher panel needs more, depending on the angle; I'd give it at least 2". A vertical panel needs at least an inch (for the diameter of the domes), but you'd probably want to leave some margin for visual borders as well.

To summarize my recommendations:

- Use the inset style
- Choose a depth based on the era your cabinet is based on:
 - For a WPC-style cabinet, inset the playfield by about 2" from the top of the wall at the front and 4-5" at the back
 - For an EM-style cabinet (1970s or earlier), inset by a uniform 1½" to 2",

or as much as needed to make room for the flasher panel



Front-to-back positioning

The second decision you have to make about TV positioning is where to put it front-to-back. Assuming you're building a cabinet to something like the standard proportions, the playfield area will be longer than a 16:9 TV, so there will be some leftover space front-to-back. The extra space typically amounts to about 6" in a standard-body cab.

The question here is whether to split the extra space between the front and back ends of the cabinet, like this:



...or to put it all at the back, like this:



In either case, some space at the back is actually nice to have, in that it's a natural place to put a flasher panel (see Chapter 56, Flashers and Strokes) or an LED matrix (Chapter 65, Addressable Light Strips).

Space at the front can also be functionally useful, because of the potential conflict between the TV and the plunger, as described in "The dreaded plunger space conflict" above. So you might have already decided to set the TV back to make room for the plunger.

Even if you're not forced to set the TV back by your plunger setup, I'd still consider it a valid aesthetic choice. Splitting the extra space front and back makes for a more balanced look, in my opinion, and I like the way an apron-like area in the front adds a 3D element.

But many cab builders are very attached to the idea of having the TV all the way at the front, so that might be a higher priority for you. For what it's worth, I also thought that way when I was planning my cab layout, and only reluctantly accepted a front gap after determining that it was the only workable solution for the plunger conflict. But as it turned out, I think I'm happier with the apron-style setup than I would have been with no front gap.

De-case it or not?

When I built my virtual cab, it was common practice to "de-case" your playfield TV, meaning that you disassembled the TV and discarded the outer plastic case, keeping only the bare LCD panel and circuit boards. The point was to remove the bulky exterior bezels around the perimeter of the screen, which at the time were often quite wide. On many TVs, the case extended a couple of inches below the bottom of the viewable screen area to make room for buttons and input jacks. That was a huge problem for cab builders, because we use the TVs in "portrait" mode - turned sideways. So if there was a wide area at the bottom of the TV, it became a wide area

along the left or right side when we turned the TV sideways. Obviously quite undesirable in a cab. The solution was to get rid of the case. After de-casing, you'd normally be left with a bare LCD panel. That still had a bezel of a sort, in the form of a metal frame holding the panel together, but it was typically fairly thin - maybe 1/2" wide - and the same on each side.

De-casing is no longer common. There are two big reasons for this.

The first is that it's simply not possible for many newer TVs. Older LCD TVs were built around self-contained panels, so you could fairly easily open up the case and extract the panel. The panel was usually a sealed unit, so it would stay in one piece when you removed it. With many newer TVs, it seems that the TV *is* the panel. That is, there's no longer anything like a separate component inside that you could call "the panel"; instead, the exterior plastic case serves as an exoskeleton that holds the parts of the panel together. If you take off the case, you're left with a bunch of loose parts that won't stay together on their own. Several people on the forums have reported discovering this the hard way.

The second reason is that there's no longer as much of a need to remove the case (even if you could). The whole motivation in the old days was to get rid of the bulky exterior bezel surrounding the viewable screen area. Newer TVs generally don't have that bulky exterior in the first place. The exterior bezels on newer TVs are usually as minimal as the interior frames were on the old de-cased panels, thanks to the exoskeleton design. Newer TVs also don't tend to have any buttons or input jacks anywhere on the front, so there's no need for one side to be any wider than the others.

I'd only consider de-casing a newer TV if you can find information on the Web about how to safely de-case *the particular model* you're using. In the absence of reliable information on the specific model, I'd plan on using the TV with its case intact. Take this into account when shopping by looking for a TV that has minimal bezels. Use the full case width stated in the spec sheet when figuring which TVs will fit in your cab. If you're designing a cab around a selected TV, figure the cab size based on the TV's case width.

Side trim

Unless you're building your cabinet to a custom width to exactly fit your TV, you'll probably have some space left over side-to-side between the TV and cabinet walls. Most people want to hide the gaps as much as possible.

The best option I know of is to use black acrylic strips, custom-cut to the required width.

You can have acrylic cut to a custom size by a local plastics store or hardware store. If you're on the west coast, check for a local TAP Plastics store.

You can also order custom plastic online at Ponoko.com. They have two drawbacks compared to a local shop: you'll have to pay for shipping, and their maximum sheet length is about 31". A TV in the standard-body size range will usually be about 36" wide. You can deal with that by splitting the trim along each side into two pieces, but that leaves a seam.

You attach the trim on top of the TV's side bezels using a strong foam tape.

Tilt-up mounting

In a real pinball machine, the playfield is mounted on a hinge at the back, so that you can tilt it up like the hood of a car.

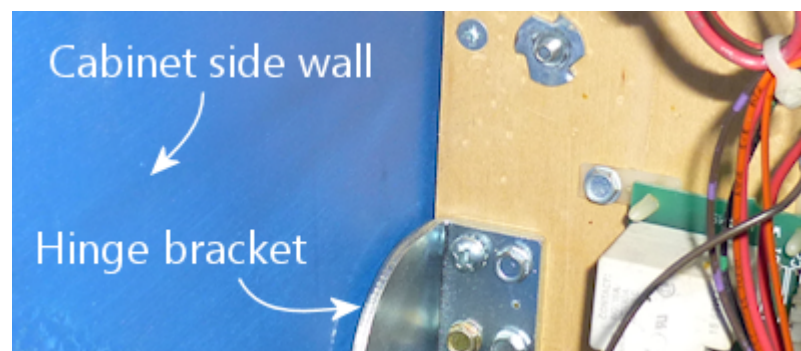


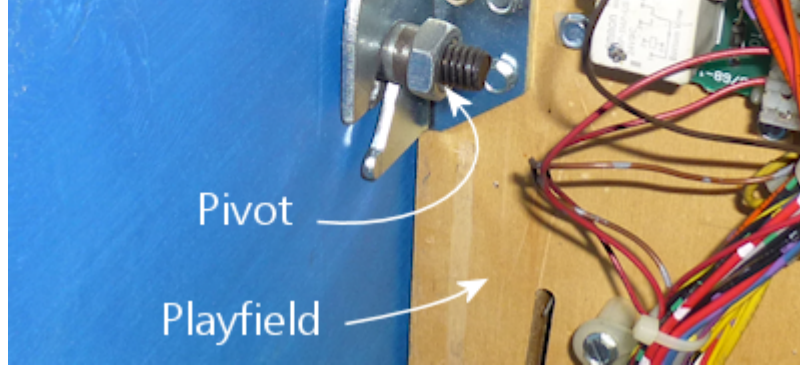
This gives you easy service access to the interior of the machine. There's nothing to disassemble, no fasteners to remove; you just take off the lockbar and lift the playfield.

The reason we're looking at how the real machines do this is that we can use it as a model for how to mount the TV in a virtual cab. Access to the interior is just as important for a virtual cab. And we can copy more than just the idea - we can adapt the mechanisms used in some of the real machines. As I've said before, it often pays to look at how the real machines accomplish things, because they came out of decades of experience solving some of the same problems.

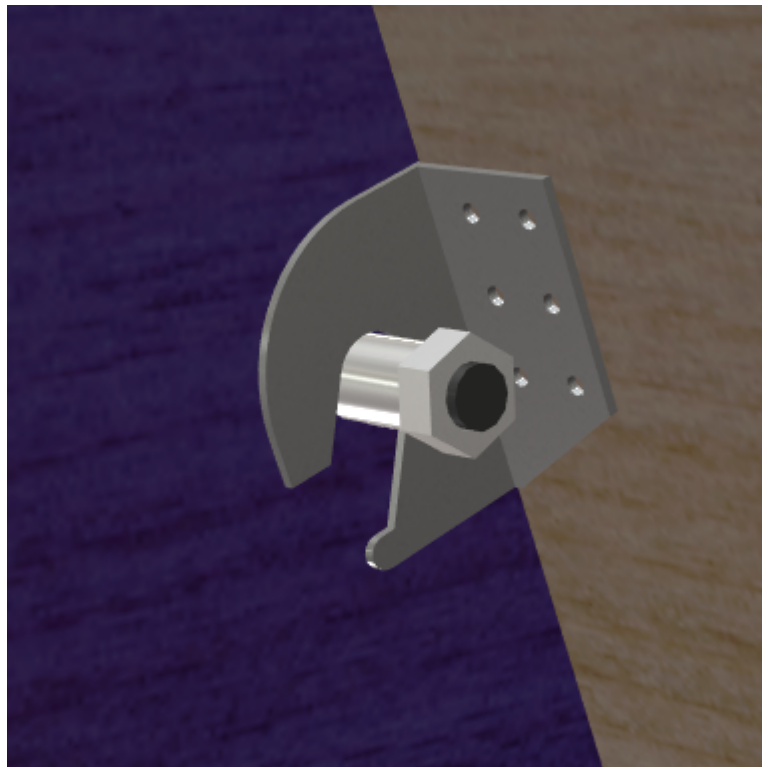
How it works in a real machine

The exact mechanism used on the real machines varies by manufacturer and vintage. The particular version that I think translates best to a virtual cab is the one used in Williams machines from the 1980s and early 1990s. They used a simple but clever scheme, with a hinge bracket attached to the bottom of the playfield, and a pivot bolt fastened to the side of the cabinet.

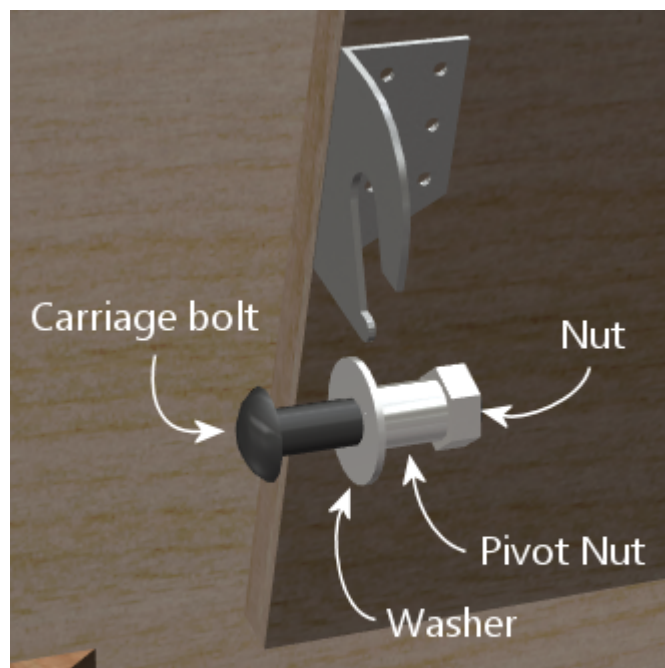




Here's a more schematic view:



Taking away the side wall of the cab for a moment, here's how this all fastens to the cab:

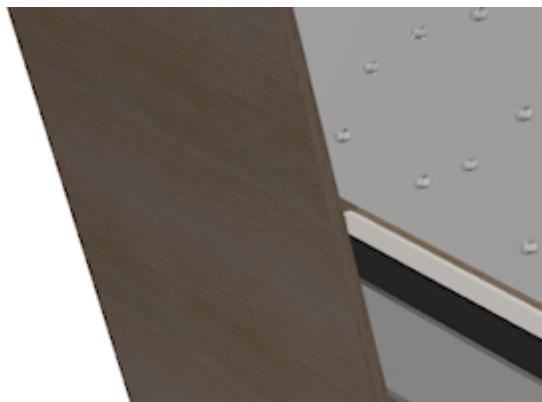
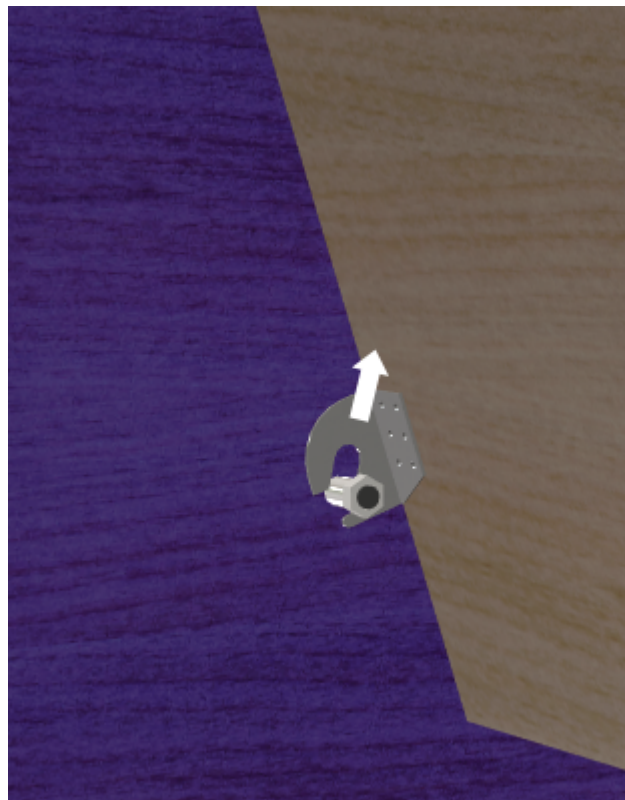




The main fastener is the carriage bolt. (That's a type of bolt with a smooth round head on the outside, without any screwdriver slots. This makes it visually inconspicuous on the outside.) On the inside, we slip a pivot nut over the bolt. The pivot nut is basically a round metal sleeve that threads onto the bolt; it provides a smooth pivot point for the bracket. A conventional hex nut is added on the end to hold lock the assembly in place.

The nice thing about using a carriage bolt as the pivot is that it only intrudes into the cabinet by about an inch. It doesn't get in the way of anything inside the cab for maintenance access.

Going back to the bracket, note how it's open at the bottom. The bracket isn't in any way permanently attached to the pivot pin, like it would be in a regular door hinge. Instead, the playfield bracket just sits on top of the pivot. It's held in place by gravity (a playfield is heavy enough that gravity does a very good job of it!). If you want to remove the playfield entirely, it's a simple matter of tilting it up like this and then lifting it straight out of the cab:





That's the really clever thing about this arrangement. With this simple mechanism, we get two levels of access, both without any tools needed:

- For routine access, just tilt up the playfield (or TV in our case) like a car hood
- If you need to remove the playfield (TV) entirely, unplug the power and video cables from the TV and lift it out

There's a surprising third benefit to this design: it's fairly cheap. Using the real pinball parts, it comes to about \$20. It would be hard to create a similarly functional mounting with generic parts from a hardware store, and even if you could, it probably wouldn't be any cheaper.

Adapting it to a TV

Here's my all-purpose plan for adapting this to a virtual cab TV. The general design should work with virtually any TV and with any cab size, although you'll have to adjust the dimensions if you're not using the standard WPC cab size.

- Create a plywood base, approximately the size of a standard playfield
- Attach the TV to the plywood base using the VESA mounting holes on the back of the TV
- Attach the pinball playfield brackets to the back of the base, just like they'd attach to a real playfield
- Attach the pivot nuts to the side walls of your cab, just like they'd attach to a real cab
- Drop the TV into the cab so that the brackets rest on the pivots, just like in a real cab

When I say this plan is "all-purpose", I mean not only that it'll fit different cab sizes, but that it'll work with any of the TV placement options we've discussed. The diagrams show the setup I like best, with a recessed TV set about midway front-to-back, with an apron at the front and flasher panel at the back. But that's all for the sake of illustration. The plan doesn't force any of those decisions on you. It's flexible enough to work with many alternative setups:

- You can put the TV anywhere you like front-to-back. The plan uses a platform that holds the TV, running most of the length of the cab. You can place the TV anywhere you like on the platform.
- You can use this plan for a flush-top TV or a recessed TV. It's just a matter of where you position the hinge pivots.

- You don't have to use an apron or flasher panel with the plan. They're optional add-ons.

At each stage in the plan, I'll point out where your TV placement design decisions come into play.

Parts

The easiest way to implement this design is with the real pinball parts. The playfield brackets in particular are highly specialized for this job; there's no generic equivalent. Fortunately, the parts aren't expensive.

- Playfield holder bracket (left side), Williams/Bally 01-8726-L-1
- Playfield holder bracket (right side), Williams/Bally 01-8726-R-1
- Pivot nut, 7/16", Williams/Bally 02-4244; or 1/2", 02-4329 (quantity 2)
- Carriage bolt, 3/8"-16 x 1-3/4", black, Williams/Bally 4322-01123-28B (quantity 2)

In addition, there are some generic hardware parts, which you can get from the pinball vendors or from a hardware store:

- Washer, 3/8" x 1" outside diameter (quantity 2)
- Hex nut, 3/8"-16 (quantity 2)

Finally, the mounting base and bolts:

- Good-quality 1/2" to 3/4" hardwood plywood (at least 2' x 4')
- M4 or M6 bolts as needed for your TV's VESA mount, 20mm length for 1/2" plywood, 30mm length for 3/4" plywood (quantity 4)
- Washers to go with the M4/M6 bolts (quantity 4)

The plywood base isn't going to be visible, but you should use high-quality material anyway, because it needs to be strong and (maybe more importantly) flat. The cheaper stuff they use for framing and roofing doesn't tend to be all that flat. You want a nice flat piece here so that the TV sits securely and doesn't wobble due to a warped base.

Don't use particle board or MDF. Particle board is terrible at supporting point weights, as we need to do at the hinges. It also tends to sag over time.

Strength and weight

The pivot setup puts all of the TV's weight on the hinges when the TV is raised, so it's reasonable to ask if the hinges are strong enough.

We know that the mechanism has a proven track record on the real machines, so as long as we're not asking more of it in terms of weight than the real machines do, we should be safe. I'd estimate that a pinball playfield (assembled) is in the range of 50 to 75 pounds.

A modern TV in the 40" range is under 20 pounds, and the plywood should be around 10 pounds. That leaves us with a weight budget of about 30 additional pounds for other features that we might want to attach - apron, flasher panel, and solenoid devices to simulate flippers and bumpers.

So I think we're very safe! The only thing to be concerned about might be a full slate of unusually heavy feedback devices. Contactors wouldn't be a problem by any means as they're quite light. Real pinball mechanisms are heavier, though (they're

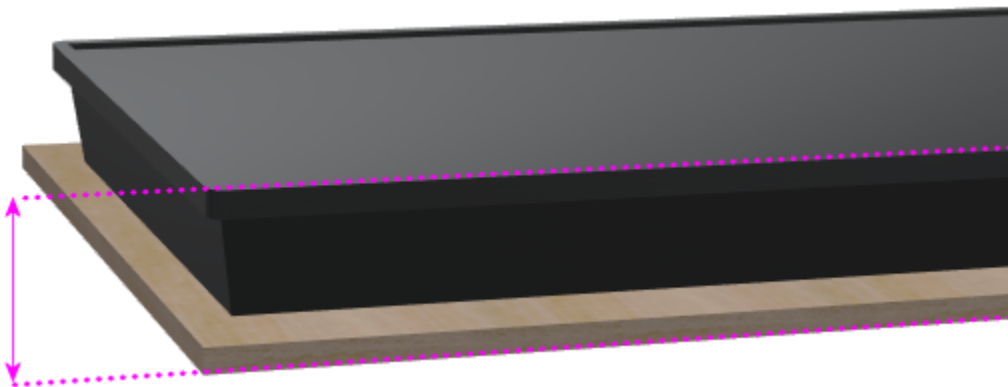
the main reason the real playfields are so heavy), so if you're using those you might want to keep track of how much weight you're adding at each stage. You can always split things up so that some of the devices are mounted to the TV platform and others are mounted in the main cabinet.

How to install

Here's the step-by-step procedure for building and installing the universal, all-purpose tilt-up mounting system.

Step 1: Measure your TV's depth. Place the TV on its back on the plywood sheet you're going to use for the base, making sure it's flush with the VESA mounting area, like it's going to be when installed. Measure the height from the bottom of the platform to the top of the TV.

This measurement will let us figure the alignment position of the platform in the cabinet that will position the TV's screen surface where you want it.



Step 2: Mark where the TV will go. Choose where you want the TV to go in the cabinet, as described earlier in this chapter. In particular, figure out the inset depth where you want the TV screen surface to lie - flush with the top of the cabinet, or set into the cabinet by some distance.

In your cabinet, measure and mark the positions where the **bottom** of the platform will go towards the front and back of the side walls. The front point should be right around the flipper buttons, and the rear point should be around the rear shelf.

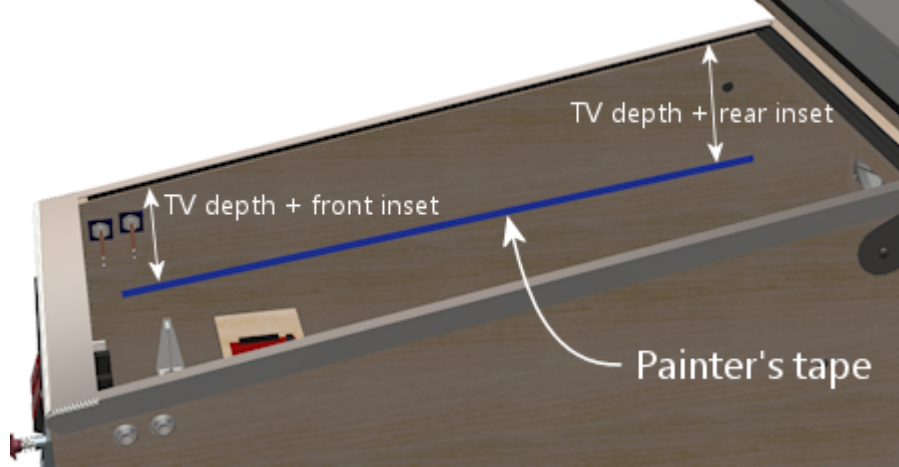
At each point, calculate the desired **TV screen inset depth** (the distance you want between the top of the cabinet and the TV screen) **plus** the TV-and-platform depth measured in Step 1.

For example, if you like my recommended inset of about 2" at the front and 4" at the back, and your combined TV-and-platform depth is 4", you'd mark a spot 6" below the top at the front and 8" below the top at the back.

Once you mark the front and rear spots, mark a straight line through the points with pencil or painter's tape. This will be the position of the bottom of the platform.

Mark both side walls the same way.



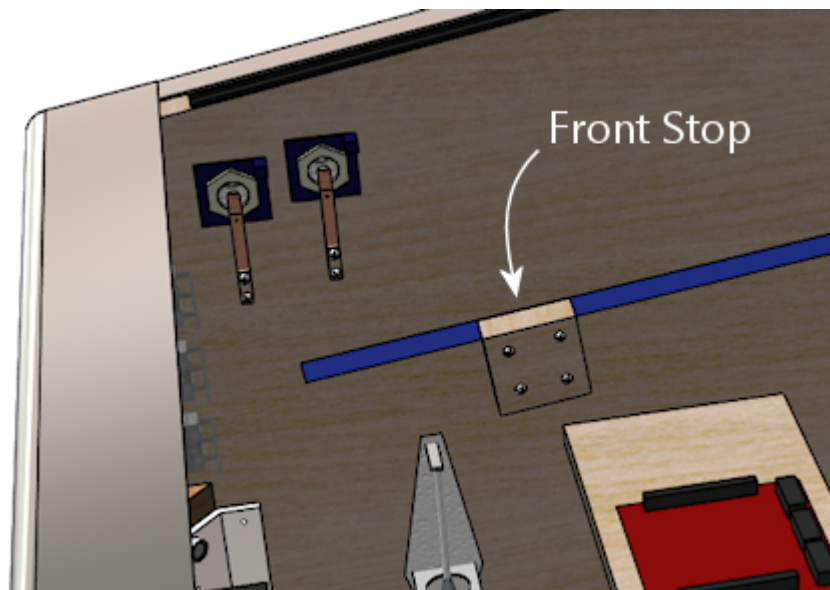


Step 3: Install the front stops. When the TV *isn't* tilted up, it needs something near the front to support its weight. I call these the "front stops" - the stops where the front of the TV rests.

Each stop will have to support about a quarter of the weight of the whole assembly (the combined weight of the TV, plywood base, and any feedback toys you attach to the bottom of the base).

You can use a sturdy metal post or a wood block for each stop. It only has to extend into the cabinet by about 3/4" of an inch to do the trick, since we're going to make the plywood platform base almost as wide as the cabinet. I'd suggest using a piece of 3/4" plywood cut to a convenient size, say 2" x 2", fastened to the cab wall with four #6 x 1 1/4" wood screws.

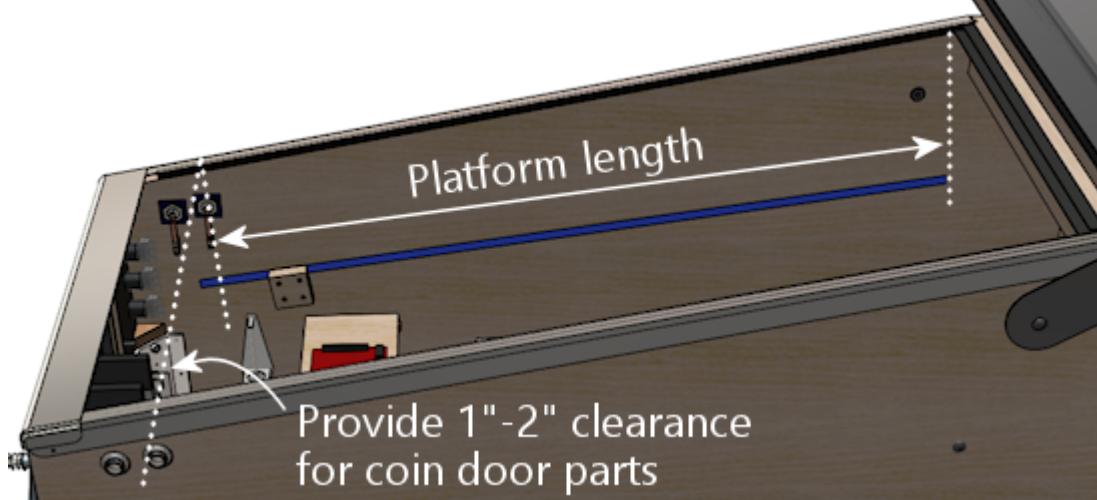
Align the top of each stop with the position where the bottom of the platform will rest, as marked in the previous step. (If you used painter's tape to mark the position, you might want to remove it before installing the block on top of it!)



Step 4: Figure the length of the base. The base should cover the area from about an inch or two behind the coin door mechanisms, to about directly underneath the backbox shelf.

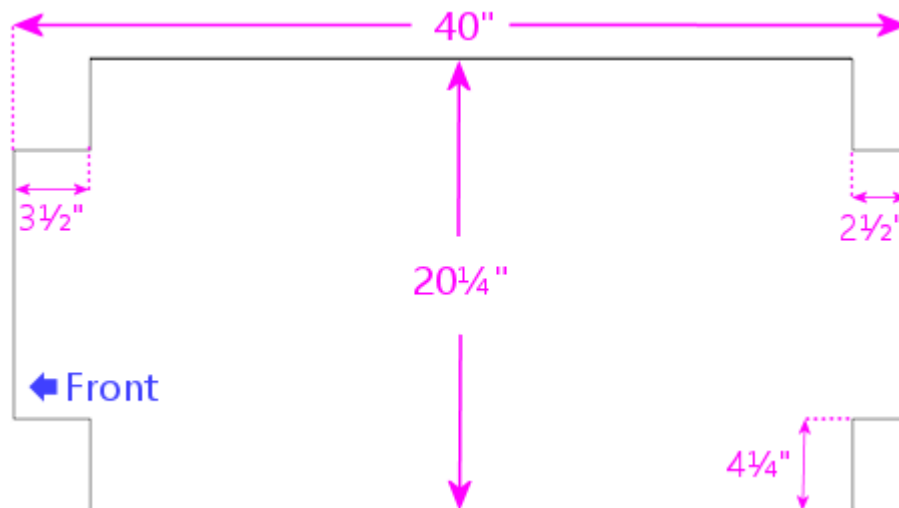
If you're using the standard-body WPC plans, the result should be about **40"**.





Step 5: Create the plywood base. Cut the plywood base sheet as shown below, making the following adjustments first:

- Adjust the width to equal $\frac{1}{4}$ " less than the **inside** width of your cabinet
- Use the overall length you calculated in the previous step.



TV platform template for a standard-body WPC cabinet. Adjust the length and width for your cabinet as described above.

The cutouts at the front are there to provide clearance around the flipper buttons and plunger. They also make it easier to mount an apron at the front, which we'll come to later. The cutouts at the back are for mounting a flasher panel.

The shape shown is only a suggestion - it's really just the simplest shape that fits the requirements. I wanted to provide something that you can use "off the shelf", but at the same time I don't want to imply that this shape is the only one that will work. Don't hesitate to adjust it to fit any special requirements of your own. Just pay attention to the core requirements that went into this design:

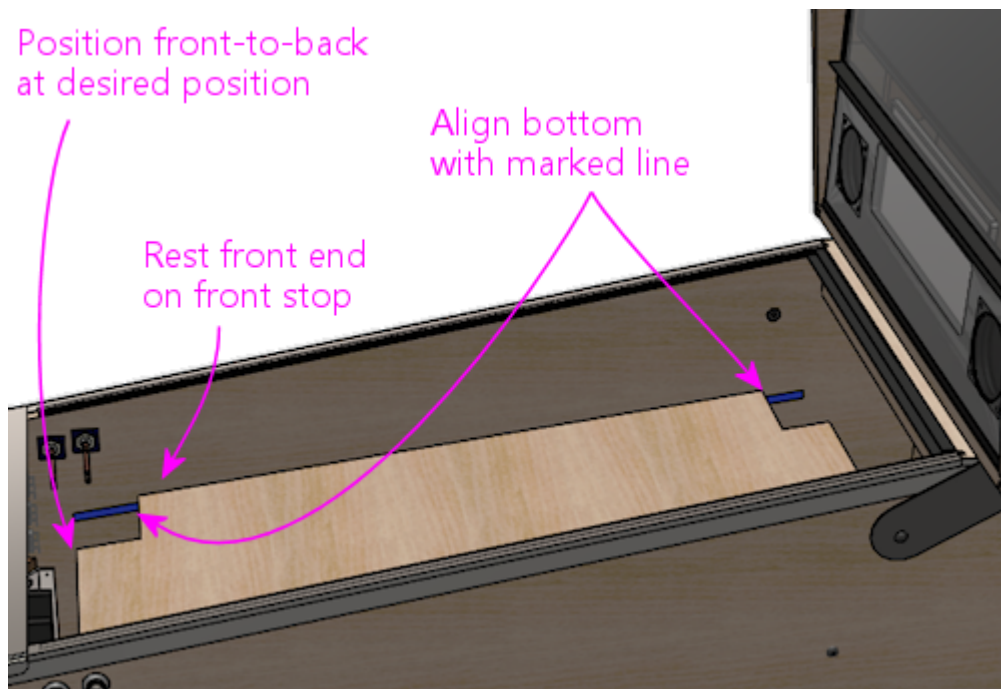
- It needs to fully cover your TV's VESA mounting area
- The front should come as close to the coin door as possible (while clearing the protrusions on the inside), so that you'll be able to reach in through the coin door and lift up the TV when you want to access the cabinet interior
- The area near the back where the hinge brackets are mounted needs to be at full width

- The area near the front where it'll rest on the front stops needs to be at full width
- Front cutouts are required to make room for the flipper buttons and plunger mechanism
- Rear cutouts aren't required, but are helpful for attaching a flasher panel

Step 6: Measure for the hinges. I'm a fan of using the actual work pieces to make the measurements whenever possible, since there's less chance of making a mistake reading the ruler, and less accumulation of rounding errors. So now that we have the platform ready, we can use it to figure where the hinges go.

This step will also give us a chance to test the fit, to make sure the platform looks as expected and fits the cabinet properly.

Place the platform in the cab where you want it to be situated when finished. Rest the front end on the front stops we installed earlier, and hold up the back end, aligning the bottom of the platform with the pencil line or painter's tape that marks where it goes.



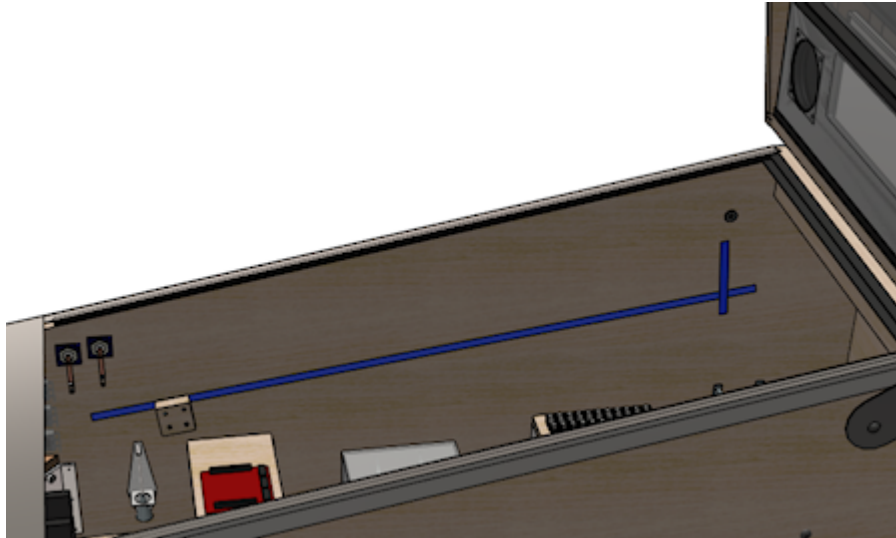
Judge the position mostly by the front: you want this to be within easy reach through the coin door, so that you can use it to lift up the TV when you want to access the interior, while leaving enough clearance that it won't collide with the coin mechanisms and other protrusions inside the door. Also check that the back lines up where expected, right around the front of the backbox shelf. Exact alignment isn't important.

Once it's in the right position, get out your pencil or painter's tape and make another mark, this time making the position of the rear cutout.





You can take the platform out, leaving behind the new rear marking.

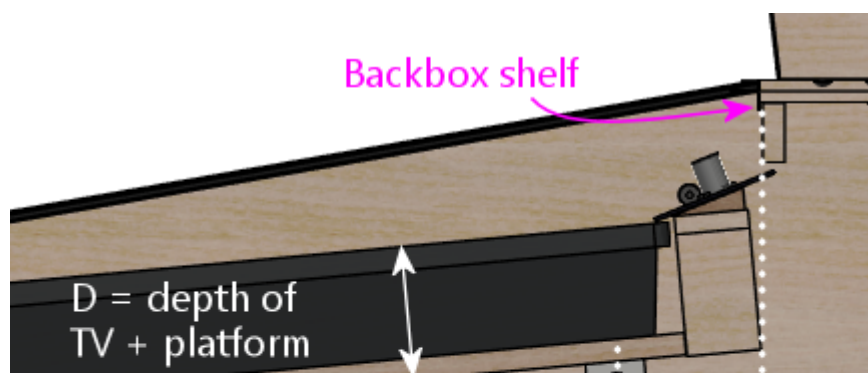


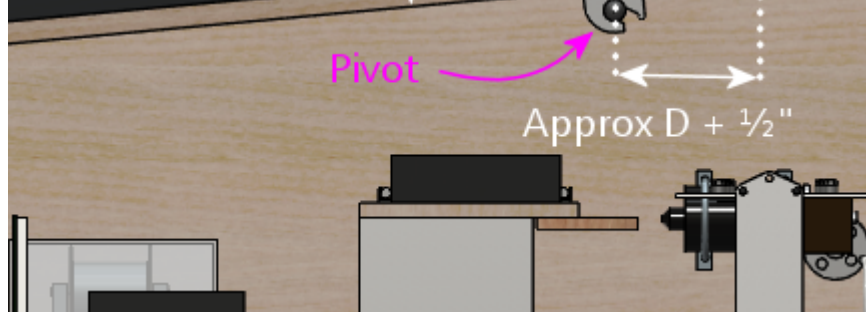
Step 7: Determine the hinge position.

Now we come to the question of exactly where to position the pivot point. It should be pretty apparent that the vertical position is purely determined by the desired TV depth. But the front-to-back position doesn't have to go at a fixed point. It has to go somewhere *near* the back to make the balance work, but beyond that, should it go at the very back, or somewhere closer to the midpoint? Remember from the picture earlier of the real pinball playfield that *they* positioned it quite a ways from the back. And they did that for a reason, which will become clear shortly.

I'm going to give you a one-size-fits-all location for the hinges, but I also want to let you know how I came up with it, and explain the trade-offs involved. You might want to check my work and figure out if you want to adjust the location for your cab.

First, the one-size-fits-all location: put the pivot point forward of the rear shelf by about the combined depth of your TV and platform, plus 1/2":



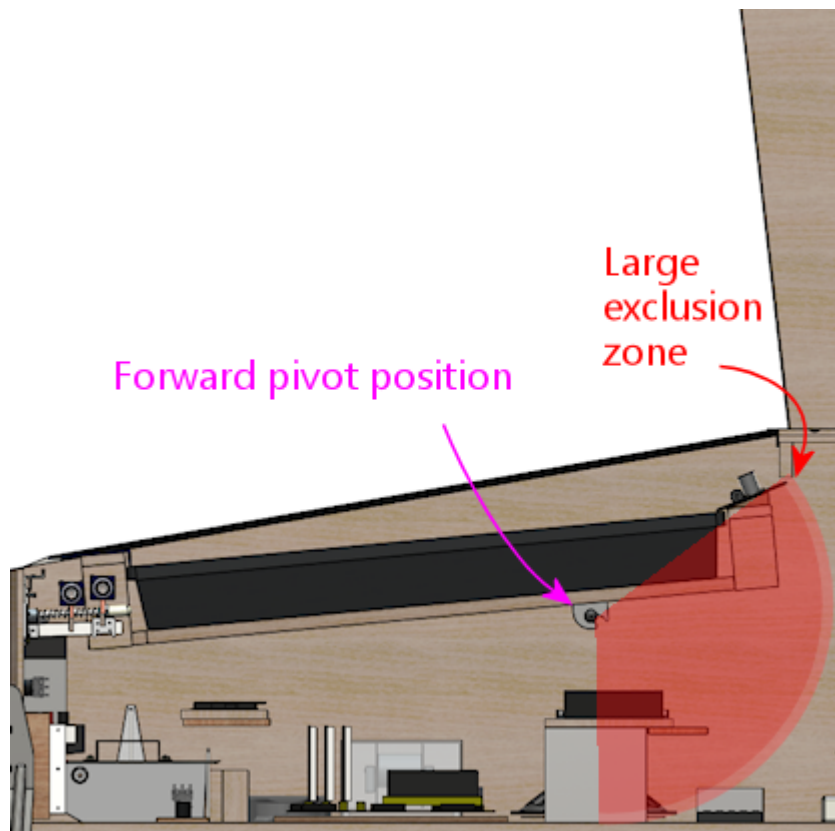


This is just a rule of thumb, so it might not be perfect for your setup. But it should be pretty good for most cabs. The reason this works is that it's just far enough forward to create clearance with the backbox shelf to allow the TV to tilt up almost to the vertical.

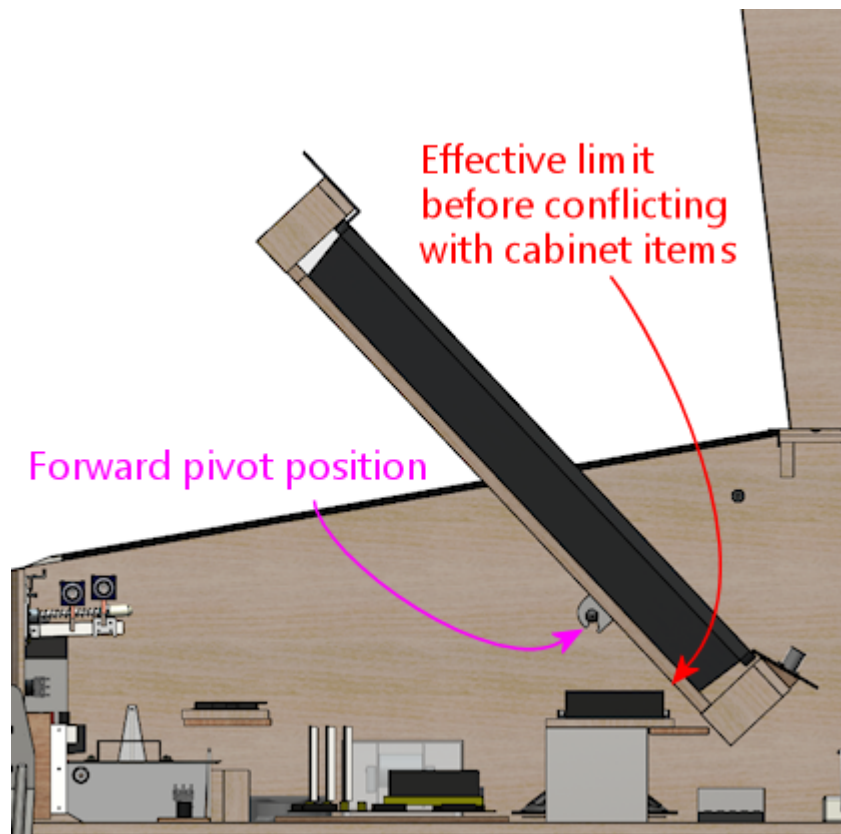
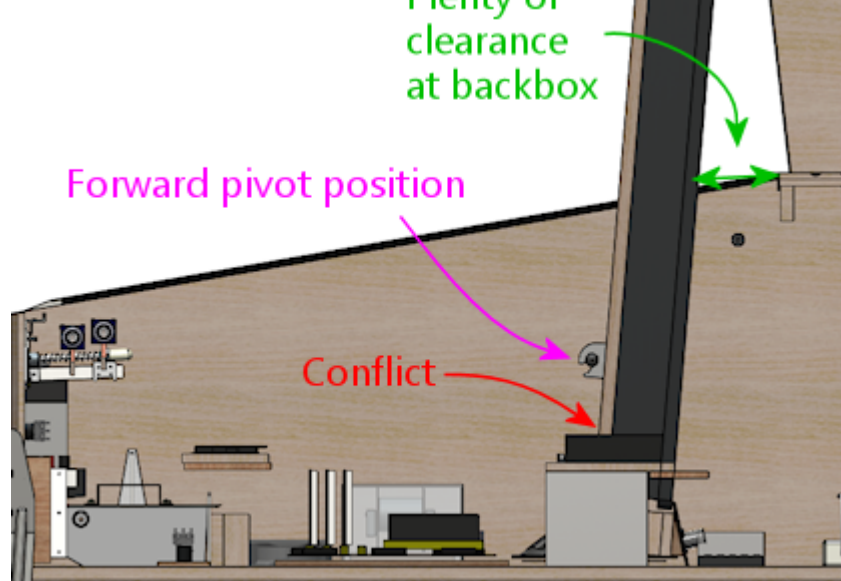
Now to the details.

The pivot point is at the nexus of some conflicting geometric constraints. On the one hand, a pivot point that's further forward in the cabinet creates more clearance between the playfield and backbox. On the other hand, the further forward the pivot point, the longer the overhang at the back that has to dip into the cabinet, which means you need more empty space within the cabinet to accommodate it.

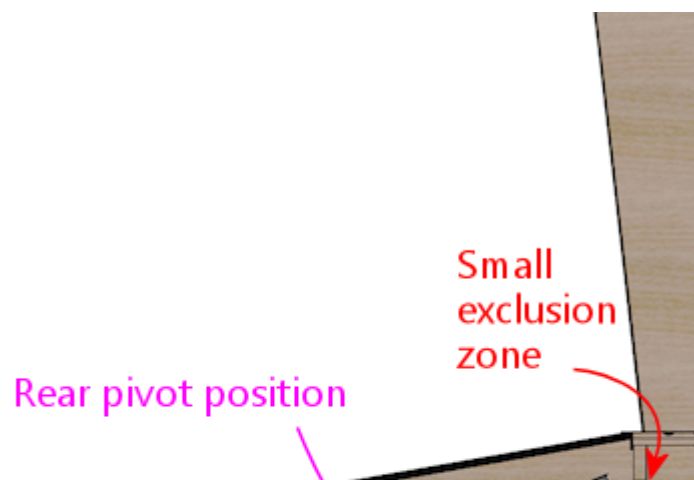
To illustrate, if we put the pivot point too far forward, we get lots of clearance above, allowing us to lift the TV more than 90°, but we create space conflicts in the cabinet below:

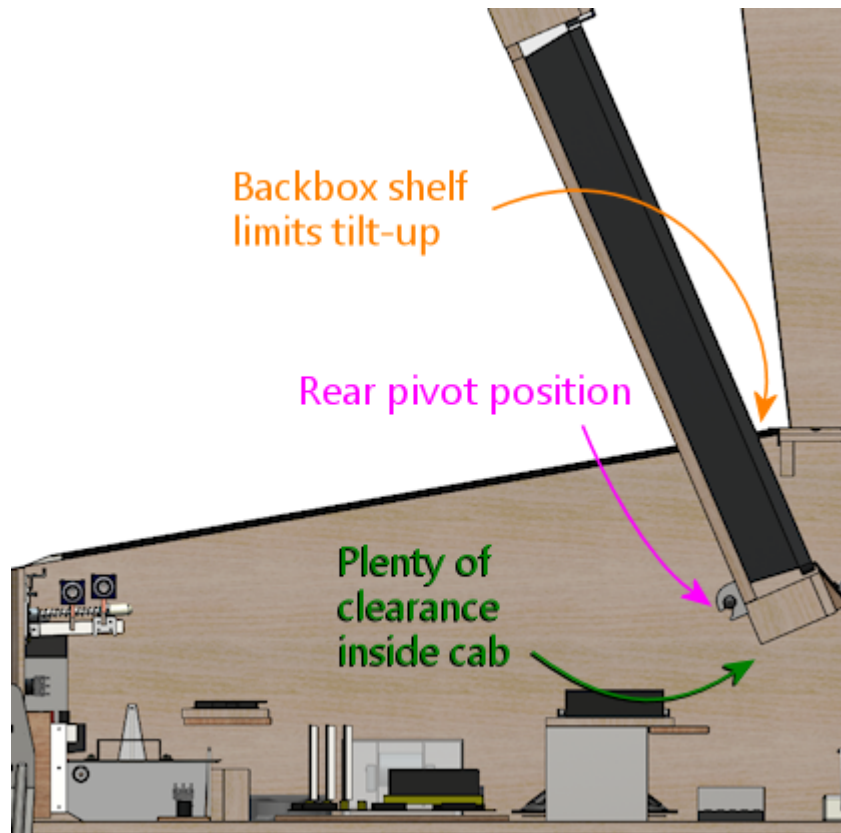


Plenty of

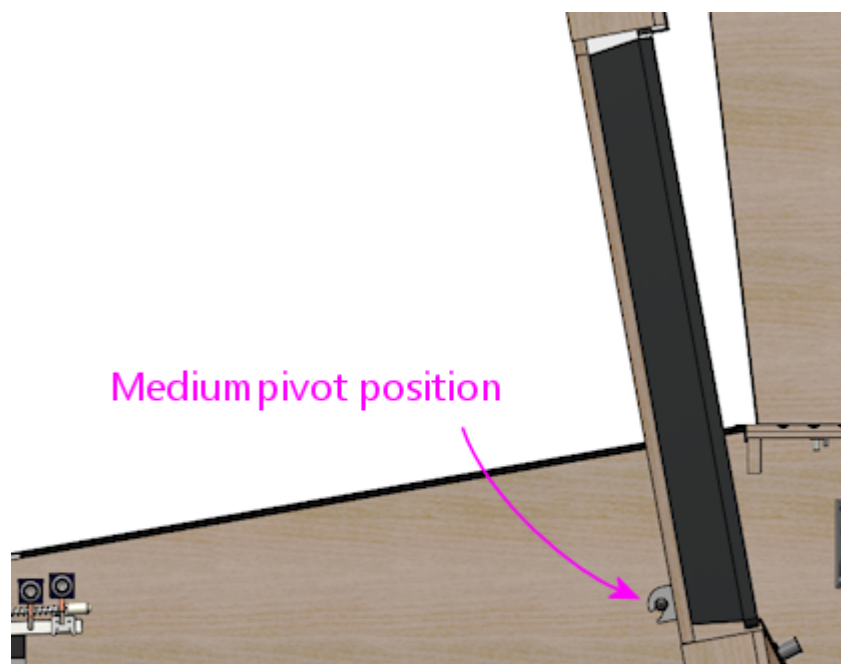


If we put the pivot point too far back, we leave plenty of room in the cabinet below, but we can't raise the TV very far before it hits the backbox shelf:





Note how the further-back position allows us to tilt the TV higher than the further-forward position when we take all of the constraints into account. But if we look between these two extremes, we can actually do even better.



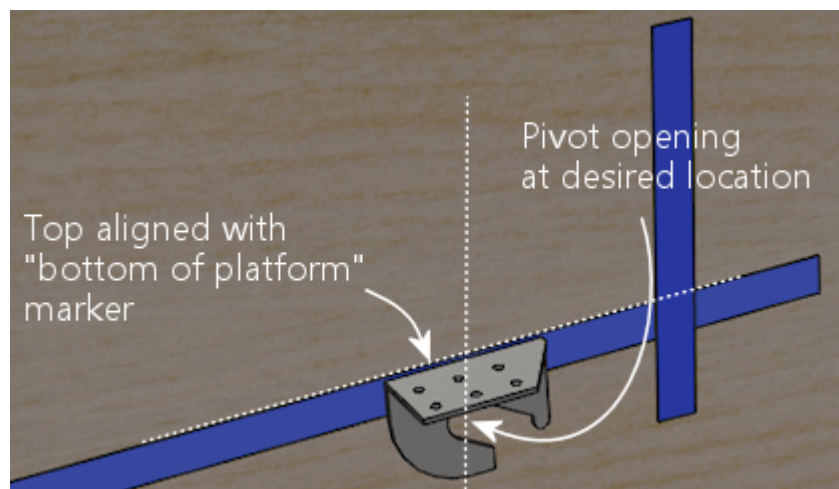


There's no solution where we're completely free of the constraints, but there's a happy medium between the two extremes where we get the best overall combined clearances, and the greatest overall tilt-up angle. That's the good news. The bad news is that I can't give you the magic optimal number for your cabinet. It's probably obvious from the diagrams that the number will be different for every combination of TV size, cab geometry, and what's installed inside the cab. The optimal number for your cab is going to be unique to your cab. So there are two ways you could approach this:

- Mock up your cab's geometry (with a cardboard model, say, or with a CAD program) and work out the optimal location by experimentation
- Pick a pretty-good-but-not-optimal location based on the rule thumb provided earlier

Even though I'm picky about these things, I think it might be just fine to go with the pretty-good solution in this case. In my own cab, I used a less sophisticated hinge mechanism that only lets me tilt up the TV by about 60°, and while that's sometimes an obstacle, it's more than adequate for most jobs. I think the pivot system I'm describing here will do better than that even if you don't optimize the pivot position perfectly - you should be able to get around 80° without trying too hard to find the perfect position. The optimal solution will be slightly better, but I think there will be diminishing returns; if the TV is in the way at 80° tilt, another another few degrees won't get it out of the way completely. And remember that this hinge mechanism also makes it easy to remove the TV entirely, so tilting up the TV isn't the last resort for the rare cases when you need unobstructed access.

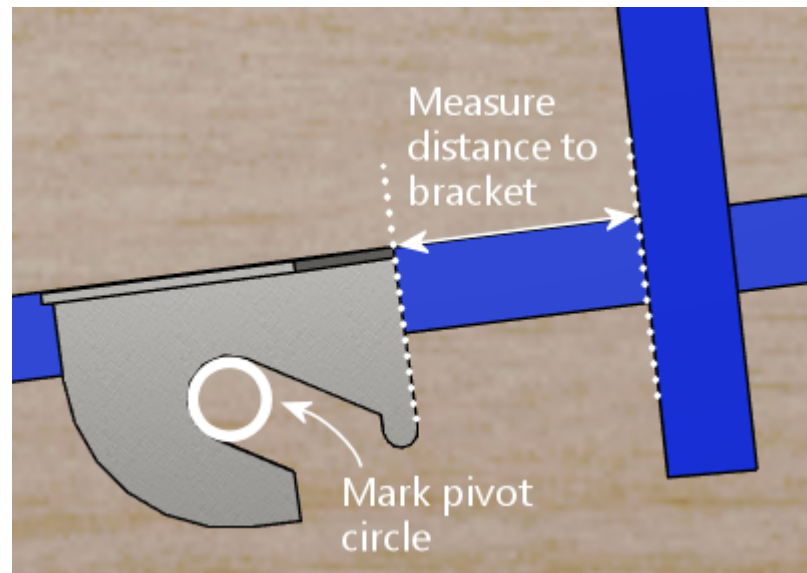
Step 8: Mark the bracket position. Using the distance to the pivot figured above, hold the bracket against the side of the cab, with its top aligned with the "bottom of the platform" line marked earlier, and the pivot opening centered on the pivot distance.



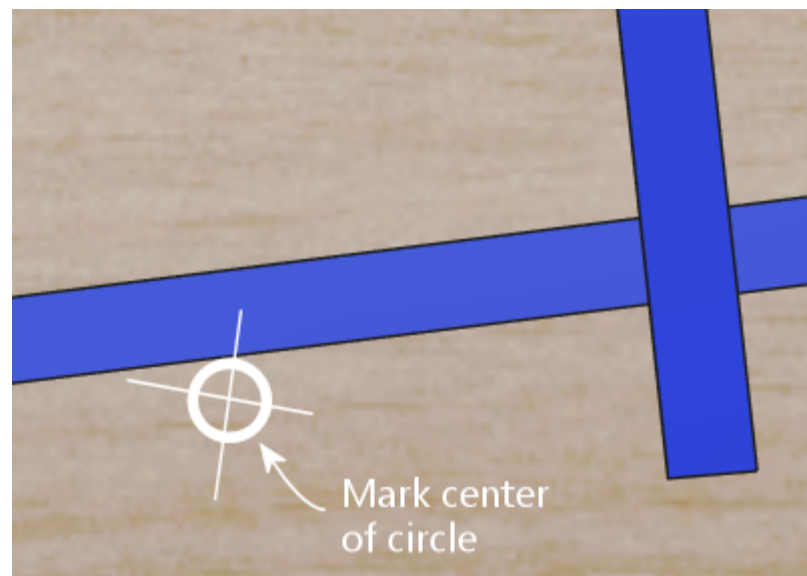
Make two measurements/markings, as illustrated below:

- Measure and record the distance between the "back of platform" and the edge of the bracket. We'll need this number when we install the bracket on the platform later.

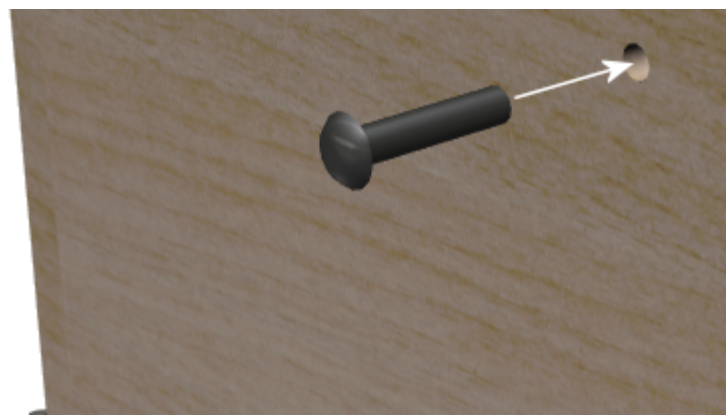
- Mark the circle where the pivot goes. We'll need this location to drill the holes for the pivot bolts.

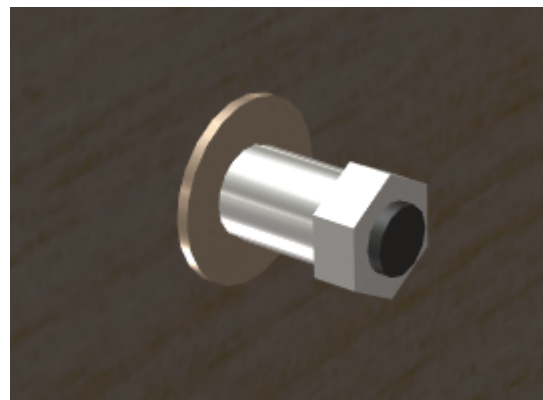
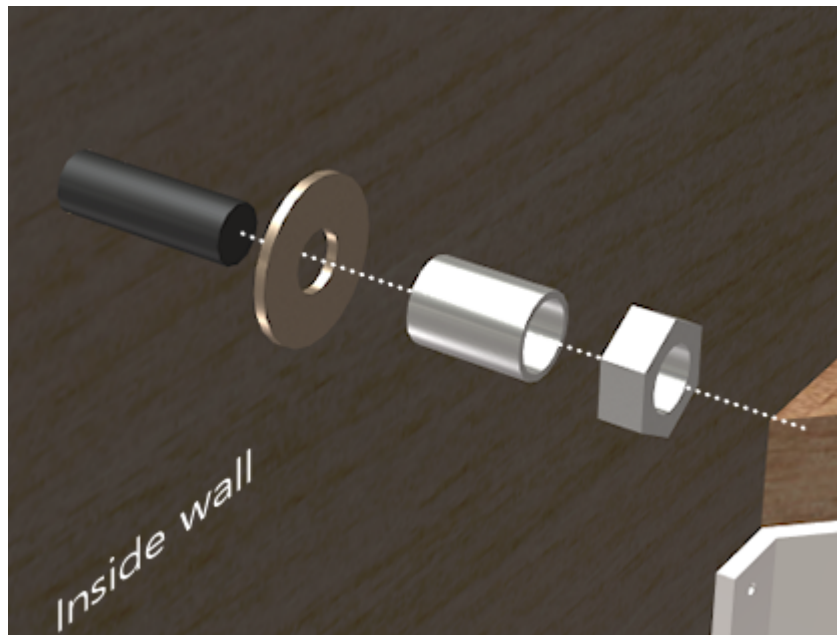


Remove the bracket and mark the center of the pivot circle. This is the drilling location for the pivot carriage bolt.



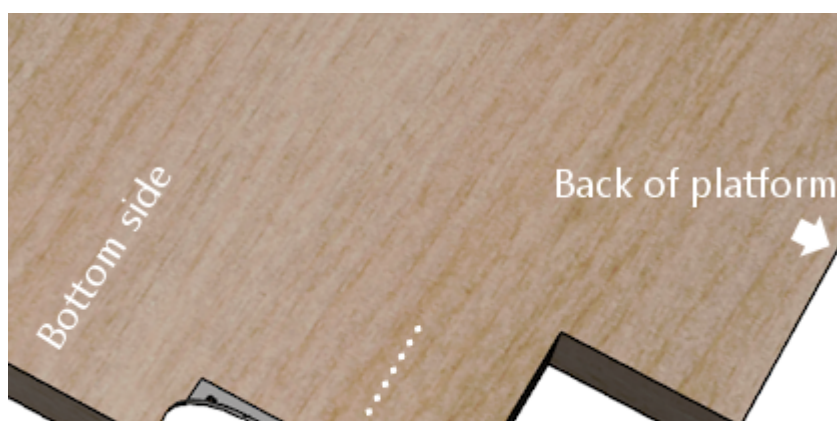
Step 9: Install the pivots. Drill a $\frac{7}{16}$ " hole in each side wall at the marked pivot position. Insert a $\frac{3}{8}$ " x $1\frac{3}{4}$ " carriage bolt into each hole from the outside. Place a 1" diameter washer over each bolt on the inside, then thread a pivot nut into each bolt and tighten. Add a hex nut and tighten.





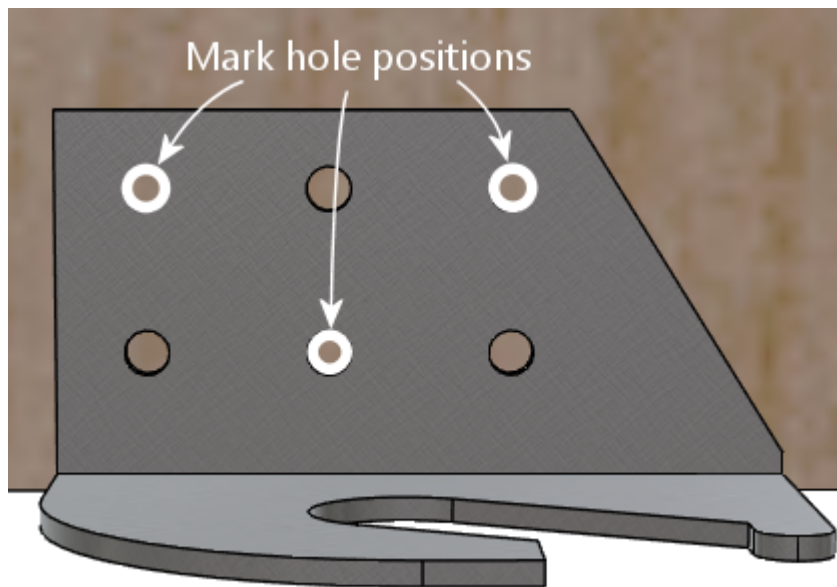
Step 10: Install the platform brackets.

Flip the platform over so that the bottom side is face up. Place the bracket onto the platform, using the "distance to bracket" that we measured and recorded in step 8, and aligning the outside edge of the bracket so that it's flush with the edge of the platform.





Mark the locations of the three holes illustrated below.



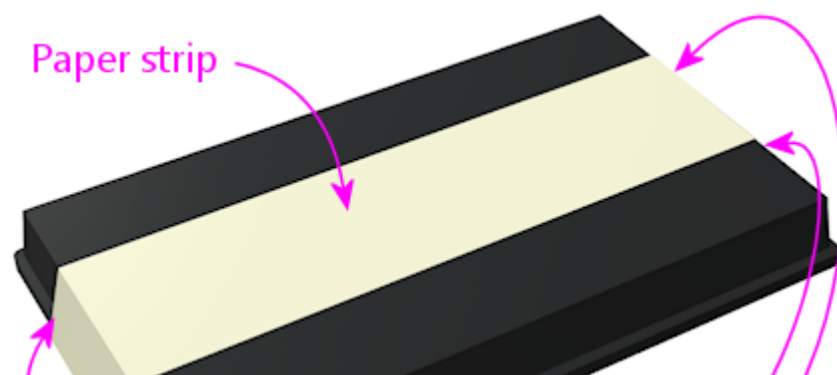
Drill holes for #6 machine screws at the marked positions.

Fasten the bracket to the base with #6 machine screws and nuts in the drilled holes. Use #6 wood screws in the remaining three holes to further strengthen the attachment.

(In terms of strength, this method of attachment should be at least as strong as on the real pinball machines I've looked at. They use two machine screws mated with T-nuts, plus four wood screws.)

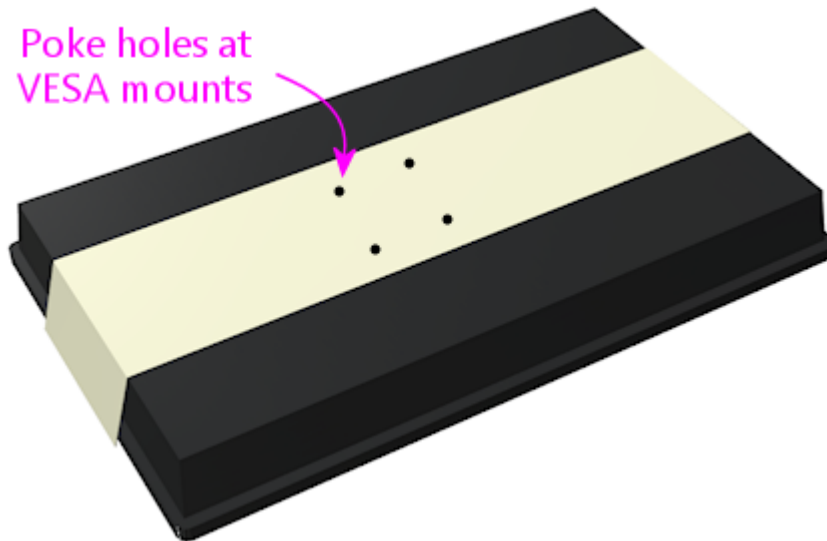
Step 11: Figure the TV position. You could figure the TV mounting position on the platform by measuring and dead-reckoning, but let me suggest a more direct approach that I think is a little easier. What we'll do is create a template for the VESA drill holes, and position the template on the platform using the TV itself itself. That will let you see exactly what it looks like in place, and fine-tune the final position.

To create the template, put the TV face-down, and stretch a strip of paper over the back of the TV, covering the VESA mount area. You can Scotch-tape together a few sheets of 8½-by-11 paper if you don't have something big enough. Use masking tape at the sides and/or front to hold the paper in place.

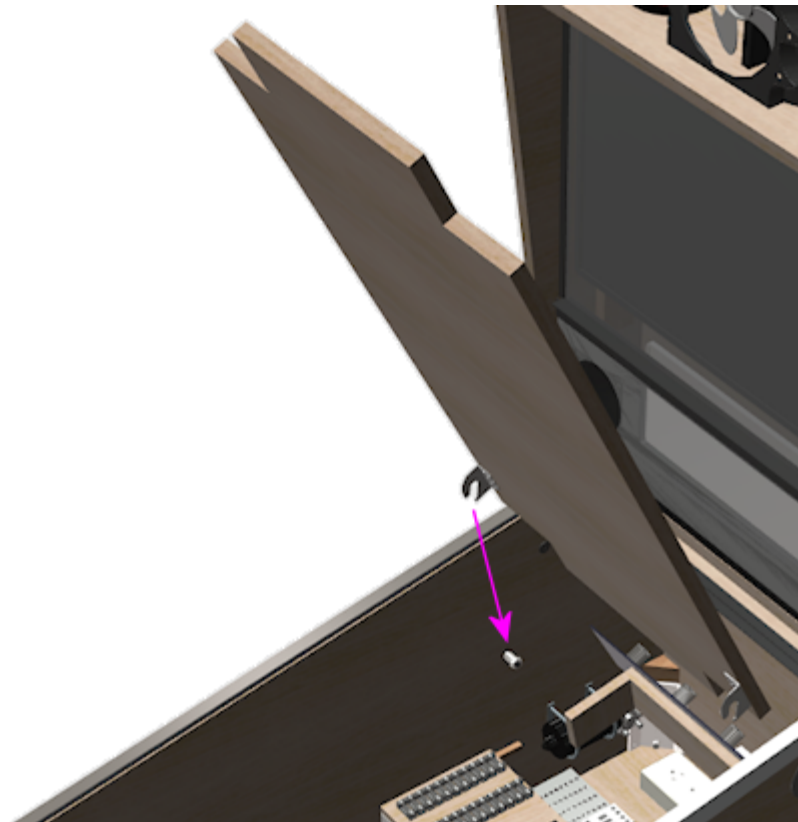


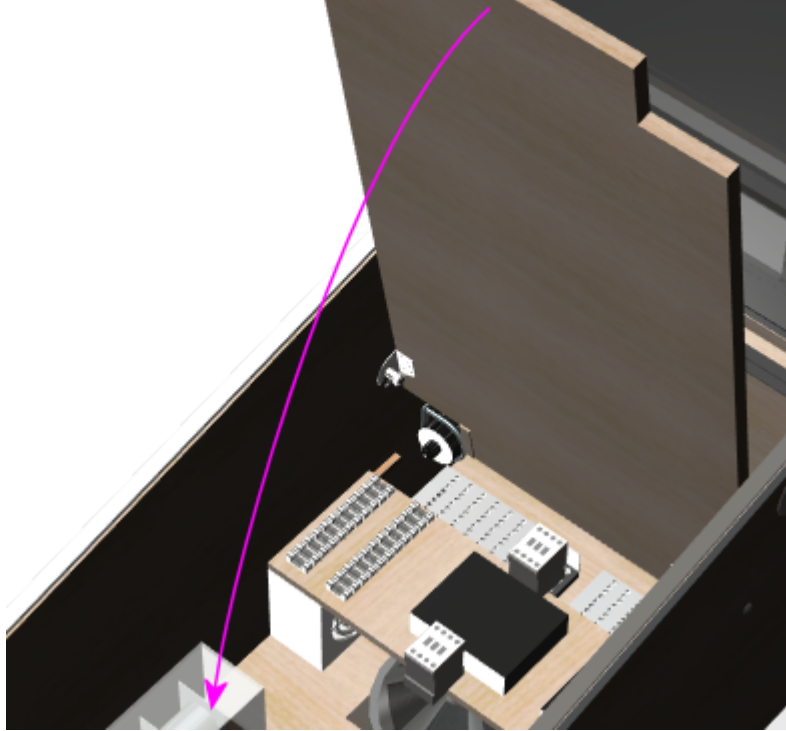


Locate the four VESA mounting holes on the back and poke holes in the paper at those spots.

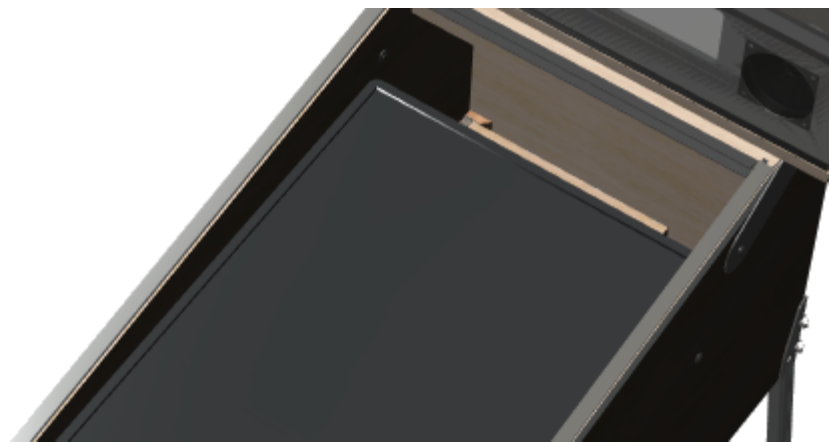


Install the platform in the cab, placing it on the hinges and lowering it to the front stops.



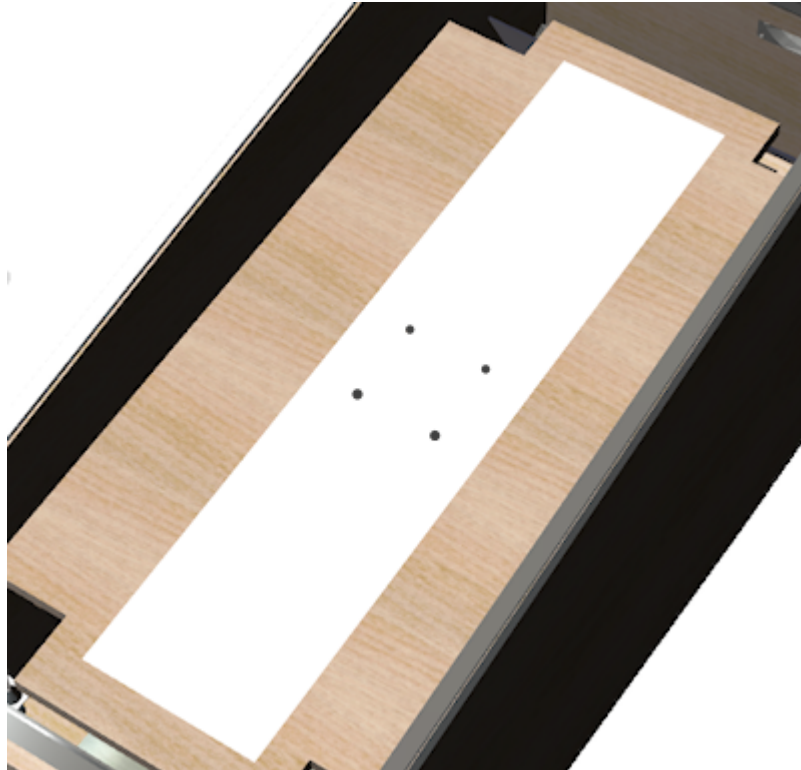


Now flip the TV over, and place it on the platform. Position it where you want it to go when this is all done.



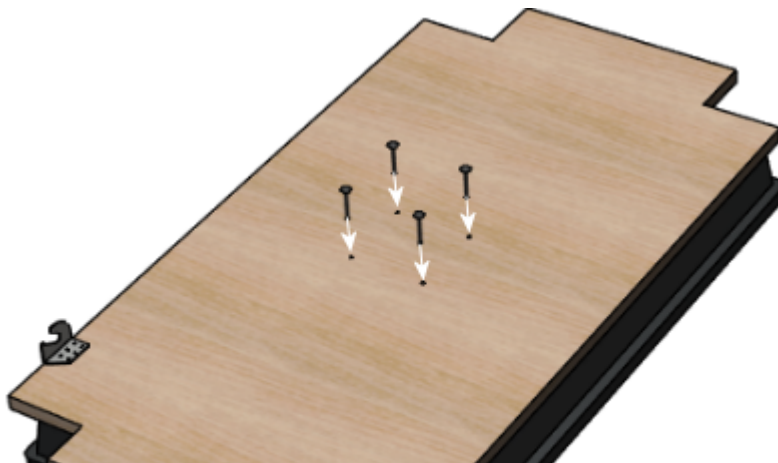


Once you're happy with the position, untape the template from the TV, and tape it to the platform instead. Be careful not to let it move at all while you're transferring it.

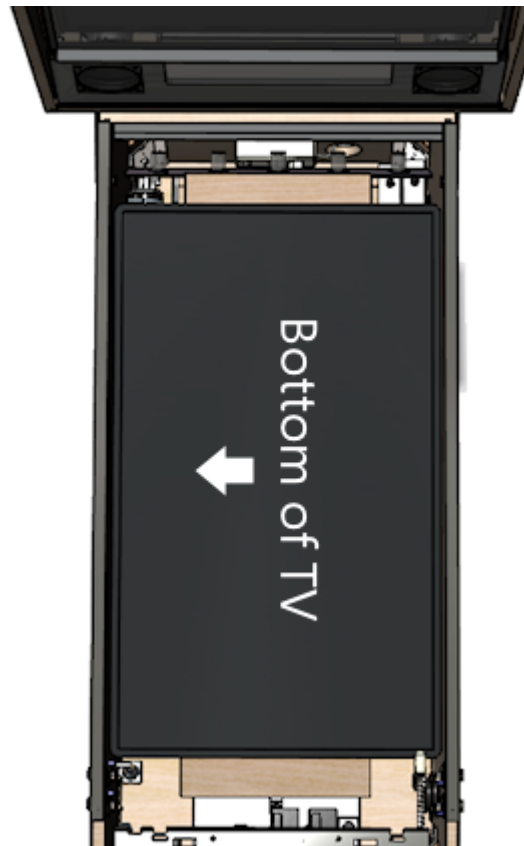


Use the holes in the paper to mark the positions of the VESA drill holes on the platform. Remove the template.

Step 12: Attach the TV. Drill holes for the VESA mounting bolts at the positions marked in the previous step. Drill 3/16" holes for M4 bolts or 1/4" for M6 bolts. Attach the TV to the base with the appropriate bolts. Use washers on the outside.

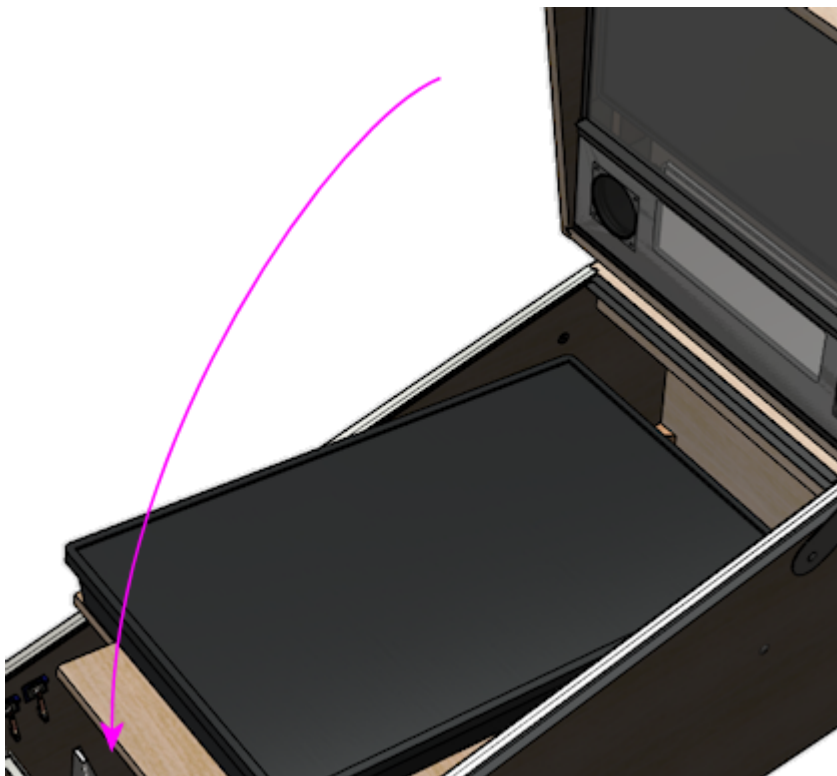
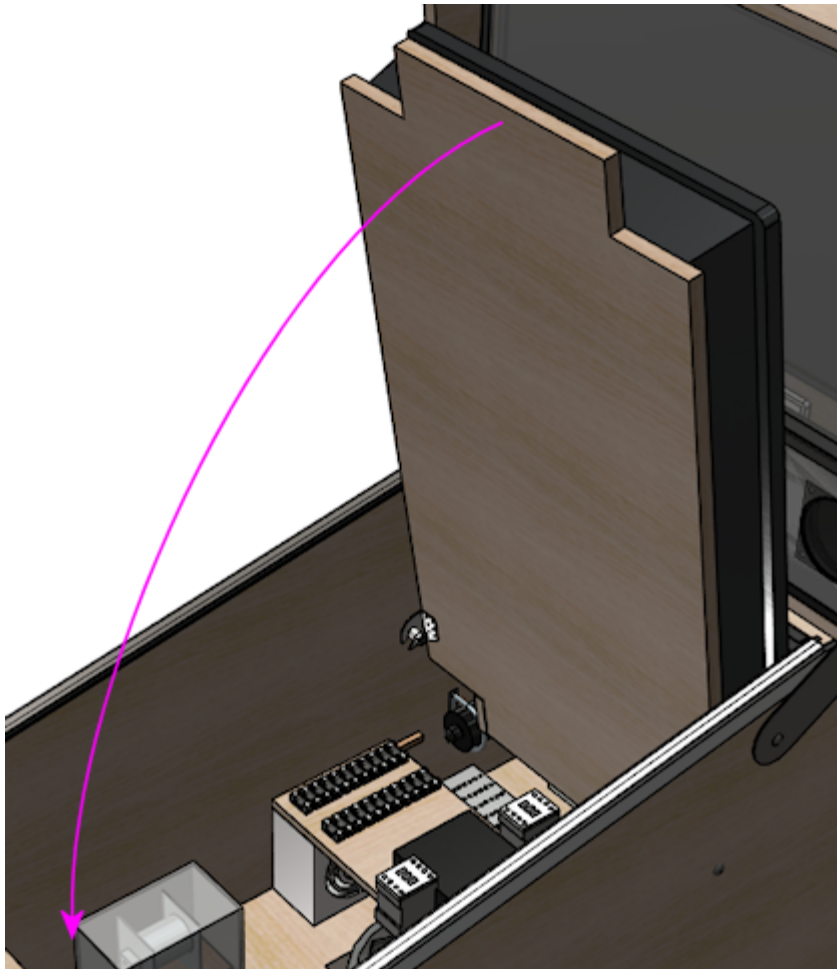
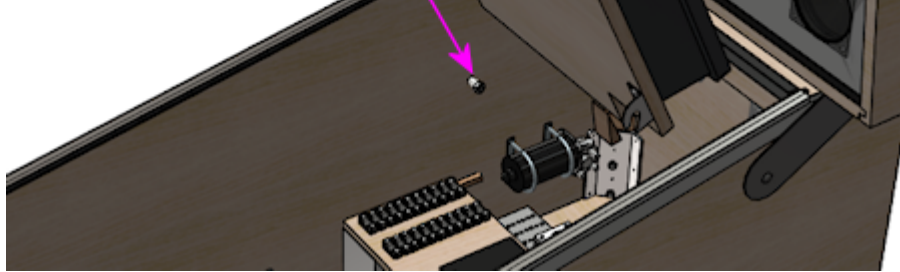


Remember that the typical orientation is with the bottom of the TV facing the left of the cabinet.



Step 13: Install and test. Hold the TV-and-platform assembly up so that it's almost vertical. Position it over the pivots in the cabinet. Lower the brackets onto the pivots. Once they're seated, lower the TV onto the front stops. Test the tilt action, checking clearances at the front and back.

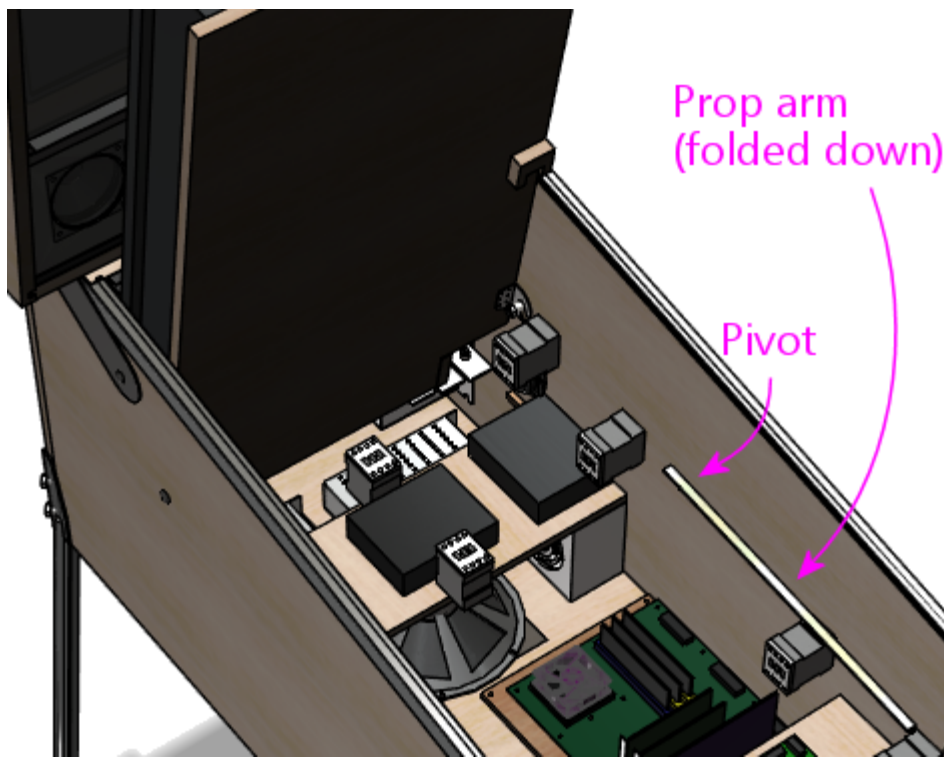


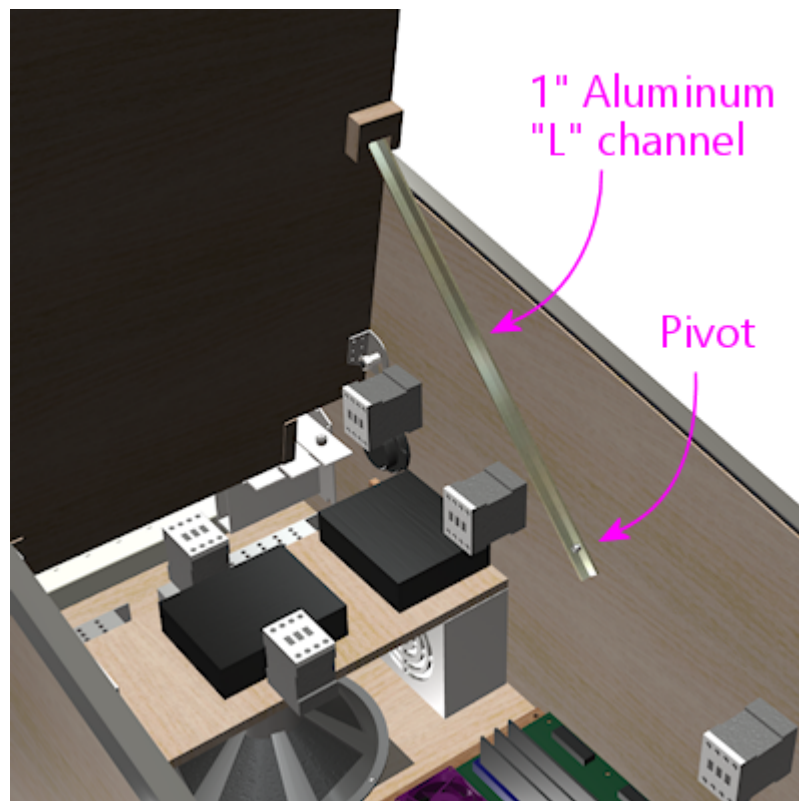
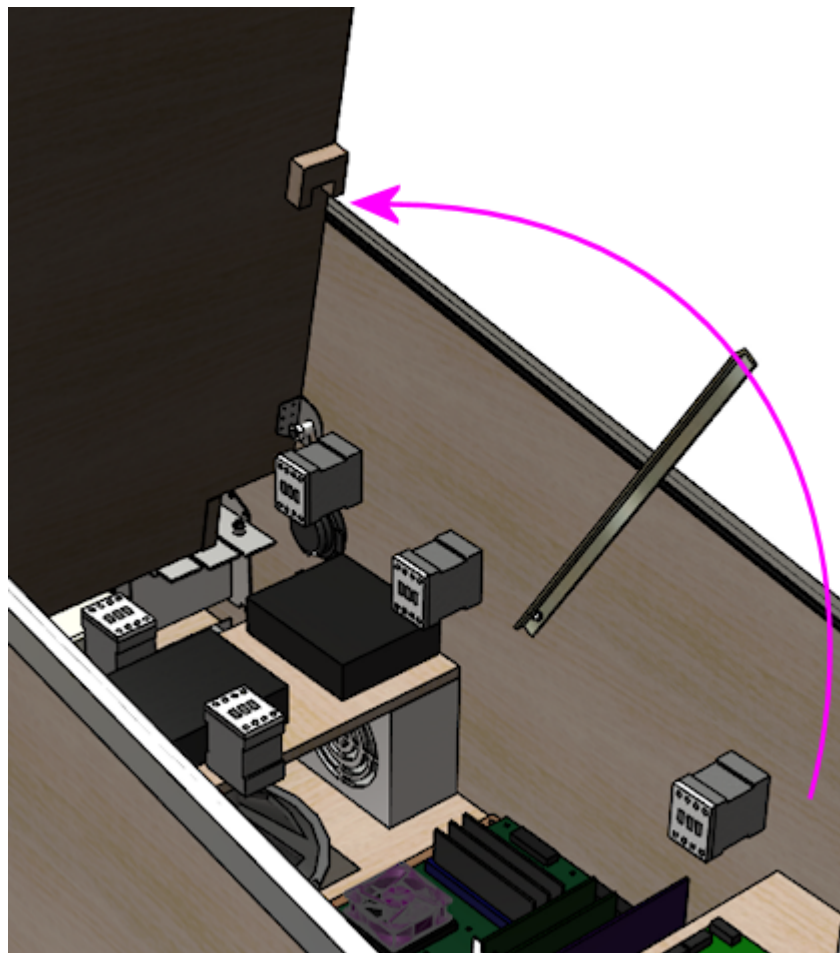




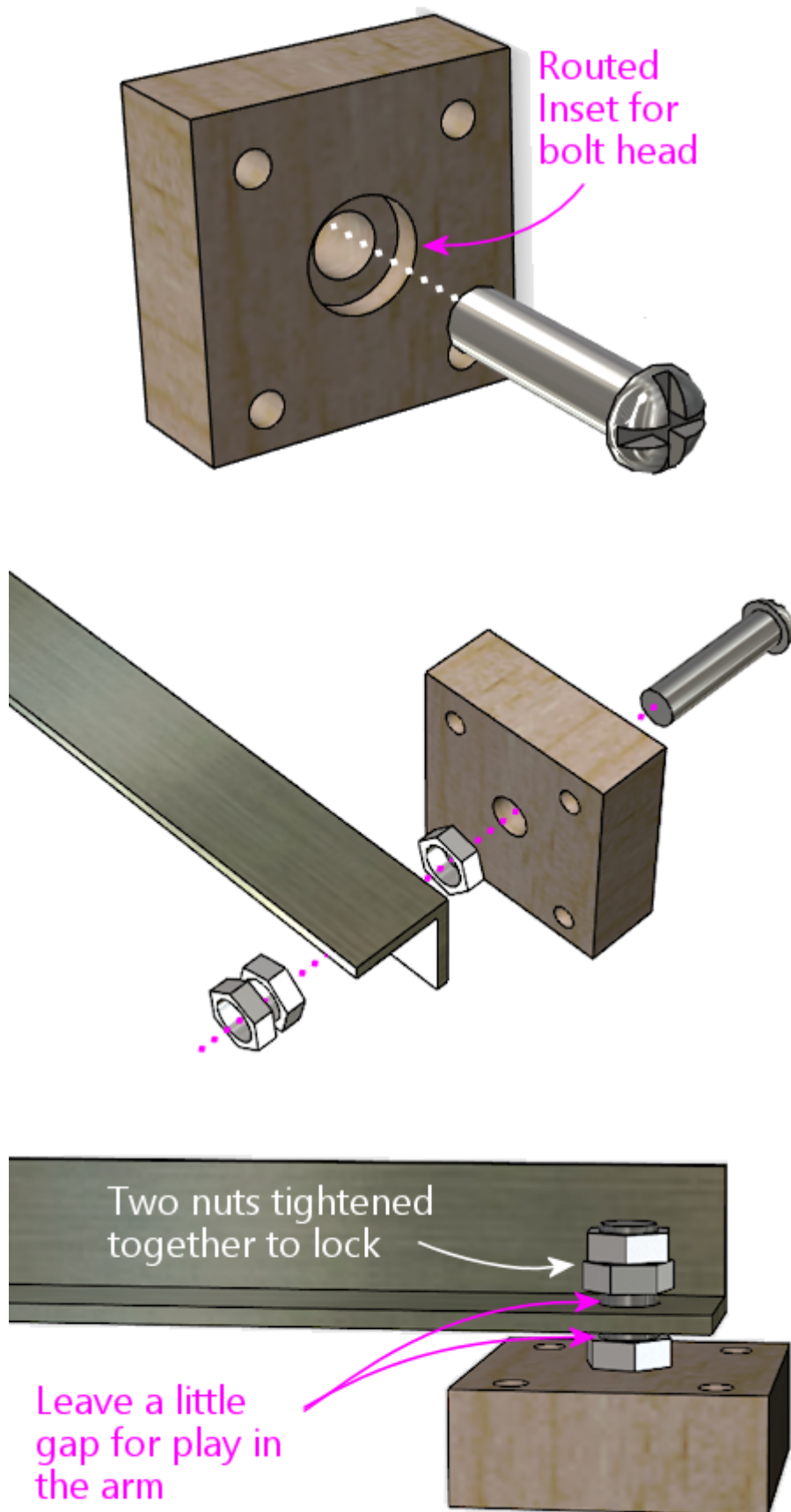
Step 14: Add something to hold the TV up. You'll need something to hold the TV in the tilted-up position when you want to work inside the cabinet.

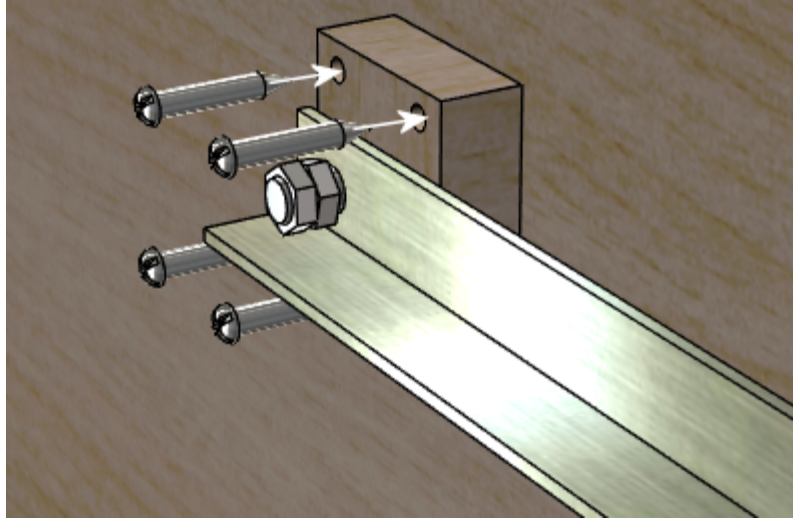
One option is to install a prop rod. The Williams System 11 and WPC machines use this approach. On my own machine, I improvised one using 1" aluminum "L" channel, cut to a suitable length. It's attached to the cabinet wall with a large bolt as the pivot.





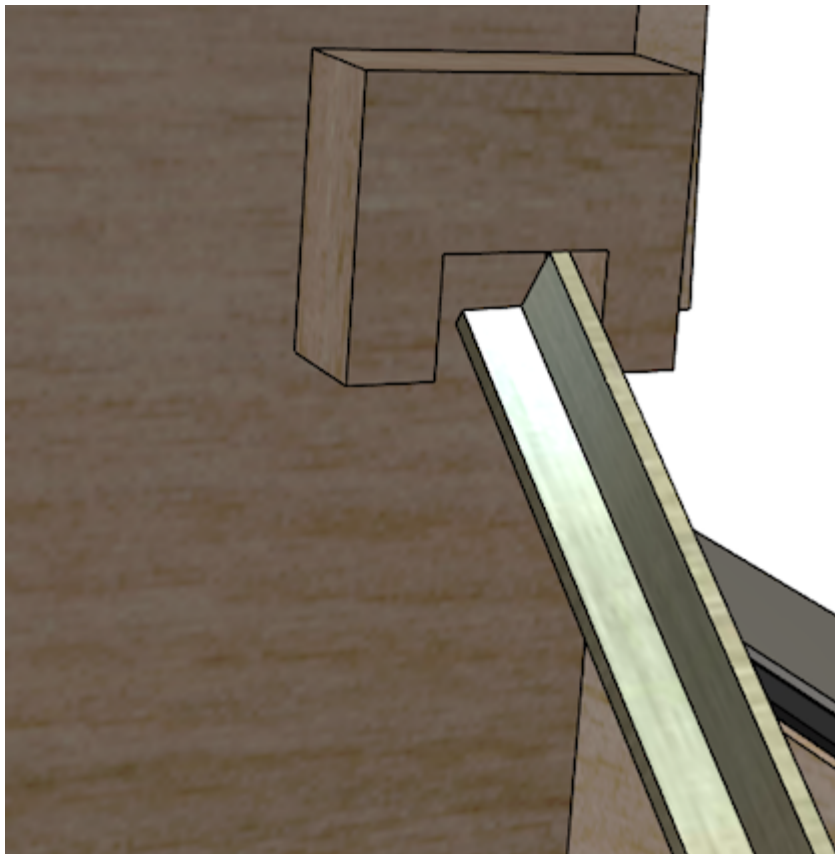
On the cabinet side, it's probably good enough to use a large wood screw (perhaps #8) screwed into the cabinet wall as the pivot. This does have to carry some weight, though, so I wanted something more robust in my own cab. I used a bolt screwed into a separate wood plate, with bolts on each side of the plate, and the plate screwed into the cabinet wall with four wood screws.





You could do the same thing more simply with a carriage bolt inserted through the side of the cab wall, if you don't mind another external bolt head adorning your artwork.

On the playfield side, I made a little wood bracket to keep it locked in place when deployed:



The prop-arm approach above has worked well on my machine, and it's not too difficult to set up. If you want a simpler approach, you could use a lanyard or luggage strap in combination with a couple of eyelets - one on the playfield, one on the backbox. Simply attach hook strap to the eyelets to hold the TV up.

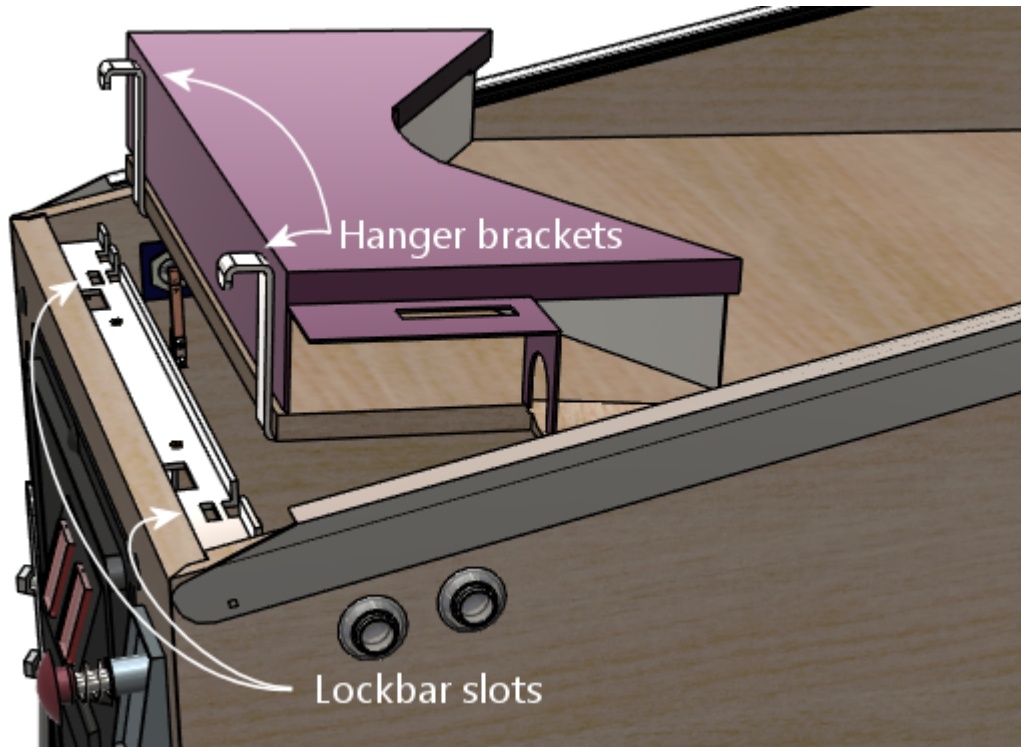
Whatever solution you use, make sure it's sturdy enough. It won't actually have to support a lot of weight most of the time, since most of the weight will be on the pivots when the TV is tilted up. But you should make it a little stronger than that, so it won't break or get dislodged if you accidentally bump the TV while working.

Hanger brackets

There's one more engineering detail on the real machines that I want to mention.

In the design above, the front end of the playfield rests on a couple of "stops" on the sides of the cab wall, as described in "Install the front stops" above.

The real machines do something a little different. They use "hanger brackets" to support the front of the playfield. These are metal hooks at the very front of the playfield that fit into slots on the lockbar:



In terms of the main job of holding up the playfield at the front, our front stops work just as well. However, the hanger brackets serve another function that our stops don't accomplish: the brackets also serve to keep the playfield from tilting *up* whenever the lockbar is in place.

Why is that important? Most of the time it's not, since gravity is enough to hold the playfield down. However, there's one situation where this changes: if you want to tip the machine up onto its back for transport. When you do that, the playfield will want to tip away from the lockbar. On the real machines, the hanger brackets will prevent it from going anywhere, since the lockbar will hold them in place. Our front stops won't do that. If you have a glass cover, the glass will stop it - assuming it's strong enough to support the weight. I'm not sure I'd want to count on that, especially if I were putting the thing on a truck.

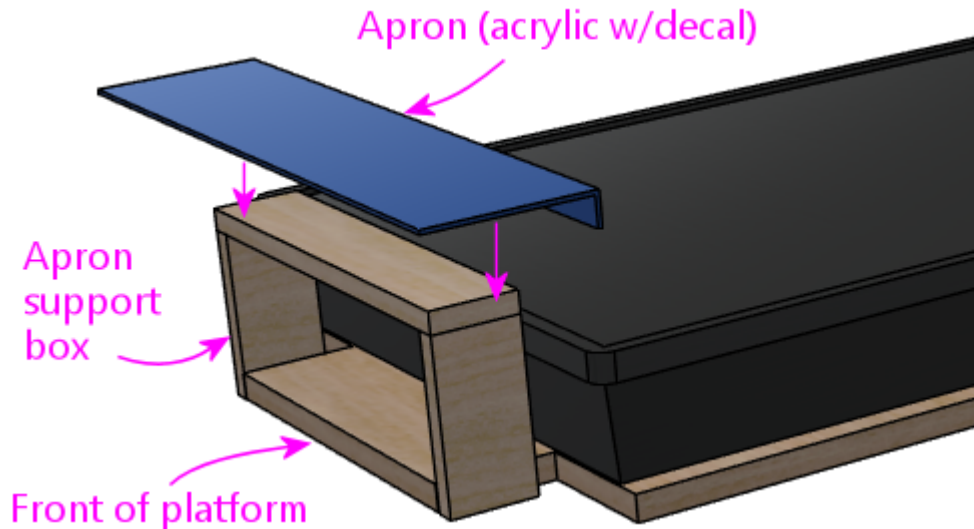
Given that we're using the standard pinball parts for the hinges, you might wonder why we didn't also use the standard hanger brackets instead of the improvised front stops. The problem is the fit. The hanger brackets are only available in certain sizes that are designed to fit real playfields. TVs are usually too deep for these to fit directly. It would be possible to adapt them with some more complex construction, but I thought the design was already complicated enough as it is.

I don't have a good alternative solution hold-down solution, unfortunately. I think the best bet if you want to ship the machine anywhere would be to simply remove the TV and box it up separately. It's some consolation that we've made it easy to remove the TV, at least!

Apron mounting

Once you have the platform assembled, it's fairly easy to add an apron equivalent, if you have unused space in front of the TV that you need to fill.

My suggestion is to build a simple box out of thin plywood. The apron doesn't have to carry any significant amount of weight, so this box doesn't have to be especially strong. Attach it at the cutouts we left at the front for the flipper buttons and plunger.



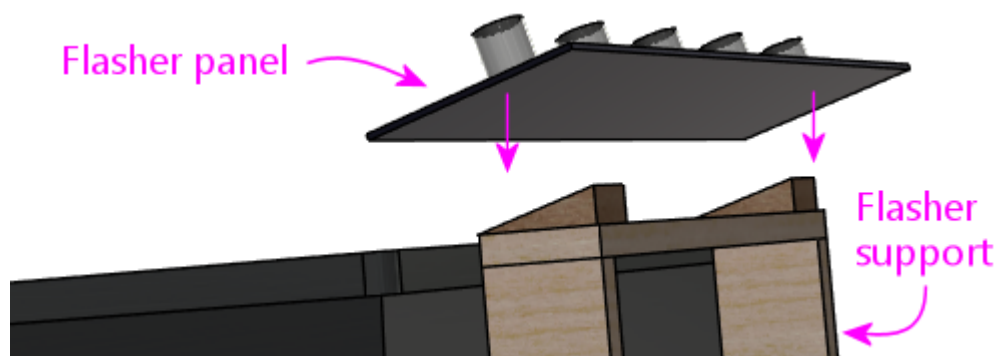
For the visible part of the apron, acrylic works nicely, since it has such a nice flat, polished surface, and it makes a great base for attaching a decal with custom graphics. You could also just use a thin plywood sheet with a nice paint finish. Attach it with whatever means are convenient, such as glue or foam tape, but I recommend Velcro to allow easy removal and replacement should you ever want to make changes.

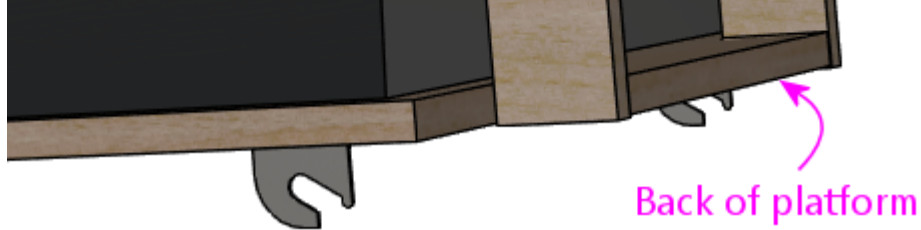
For ideas about designing the apron's cosmetics, see "Apron" in Chapter 43, *Finishing Touches*. That section includes a template for laser-cutting an acrylic cover with cutouts for standard-sized pinball instruction cards.

On the real machines, the apron sits well above the playfield, usually 2 to 3 inches. You probably don't want quite that much depth on a virtual cab; this is more about creating an impression than exact duplication. I think a vertical distance of about 1" or a little less looks good. On the other end, the apron should be set in a little from the top glass as well, perhaps another 1" on that side.

Flasher panel mounting

Adding a flasher panel at the back is basically the same as adding an apron at the front. Build a little box to serve as the platform, and mount the panel on top of that.

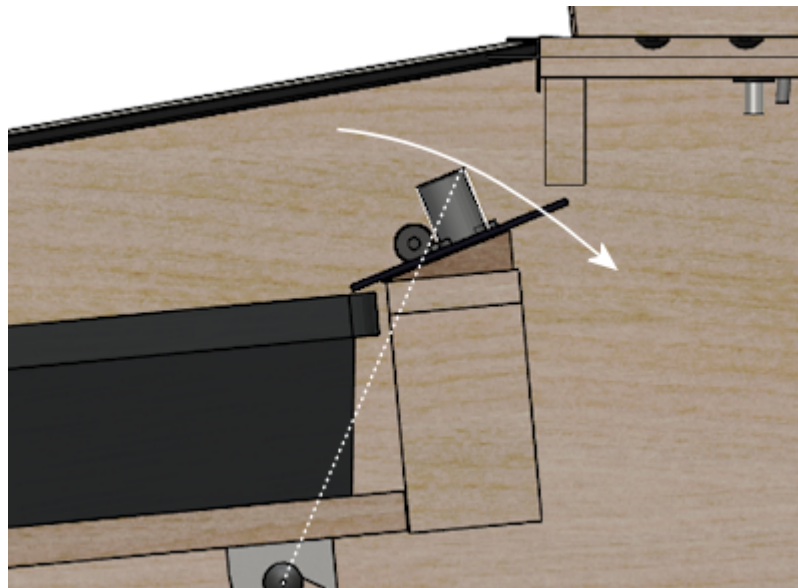
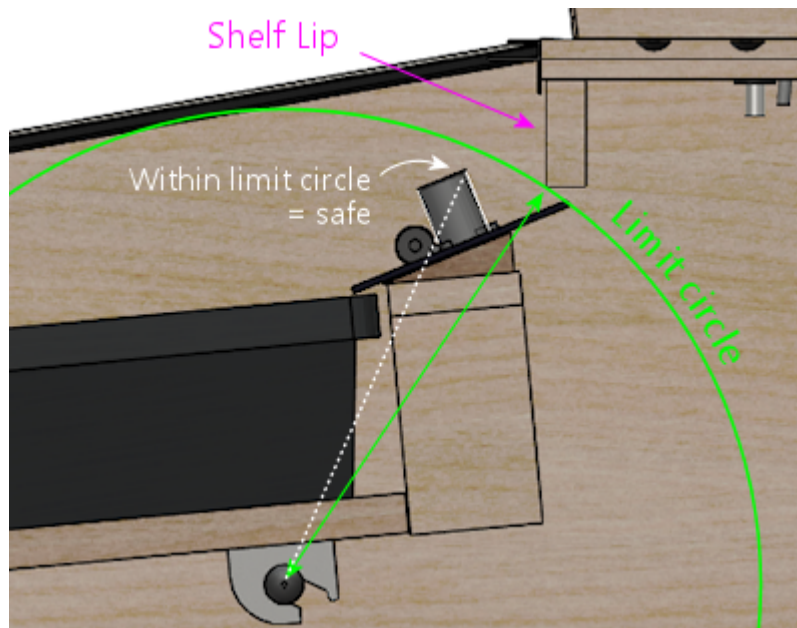


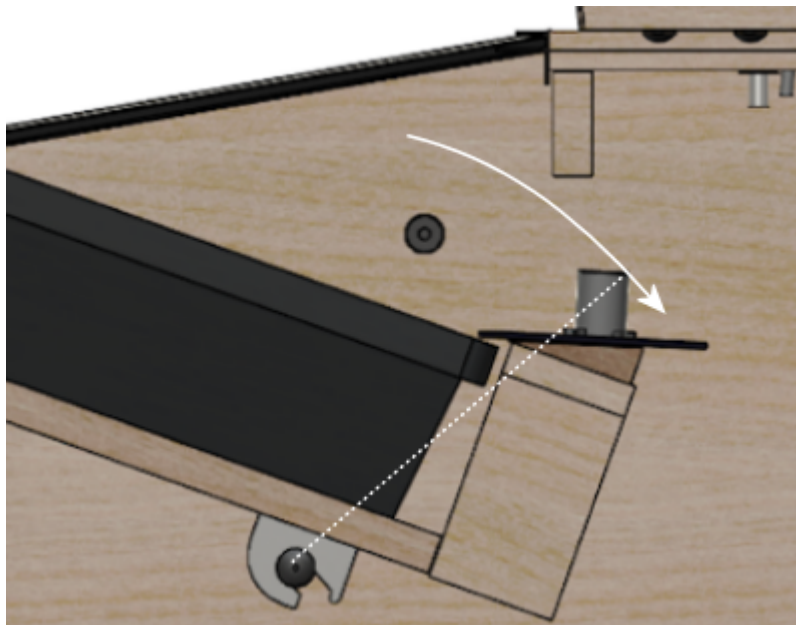
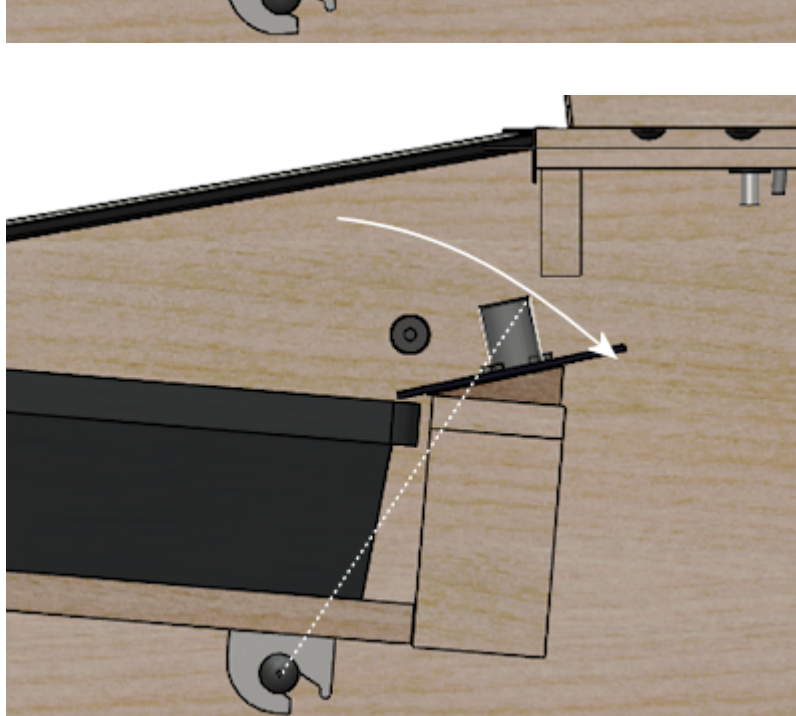


See Chapter 56, Flashers and Strobes for more on designing and building the flasher panel itself, including the electronics and how to connect it to the software.

One extra detail that you have to pay attention to with the flasher panel is clearance with the lip below the backbox shelf. Depending on how your panel is set up, the domes might stick up above the bottom of the lip. If they stick up too far, they might hit the lip when you lift the TV.

The easiest way to be sure is to test it, but you can also figure it out from the measurements when planning. The key is that everything rotates around the pivot point, so everything always stays at exactly the same distance from the pivot point - that is, everything moves along a circle centered at the pivot. That means that the flasher domes won't collide with the lip as long as the highest point on each dome is within the limit circle of the shelf lip:





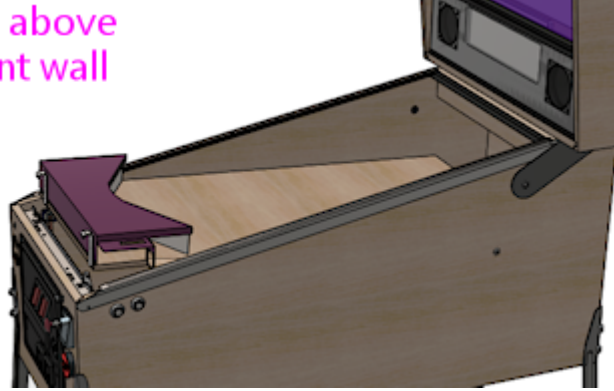
Alternative tilt-up design with a sliding pivot

The later WPC machines and modern Stern machines use a different, more elaborate design that I at least want to mention, even though I don't think it translates well to a virtual cab.

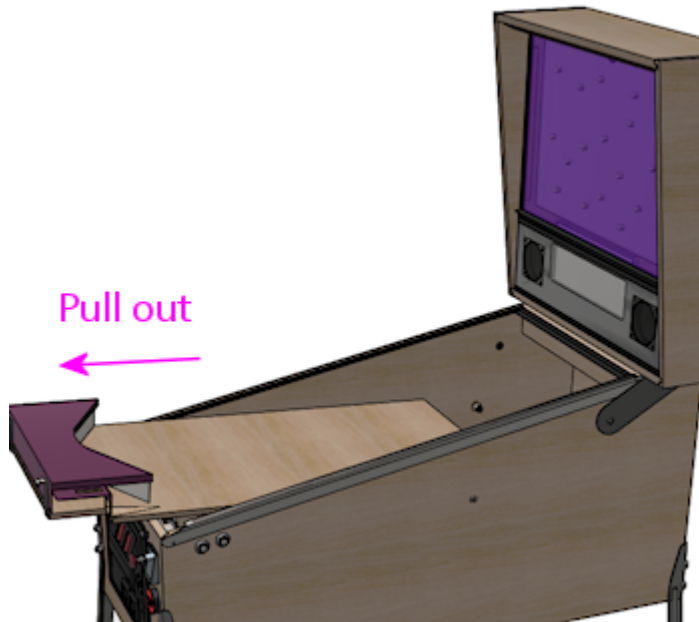
The newer machines use a sliding pivot point that lets the playfield *slide forward* before tilting up. They switched to this new system because it provides more vertical clearance than the old fixed-hinge system, allowing for longer playfields and taller ramps.



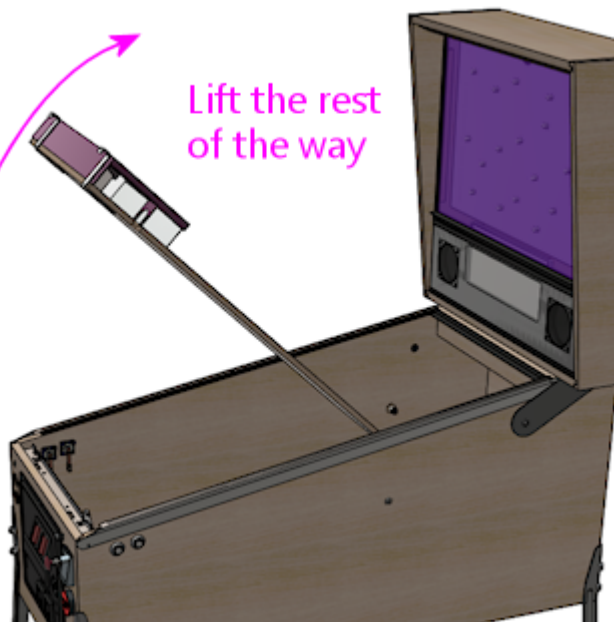
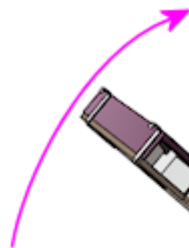
Lift above
front wall



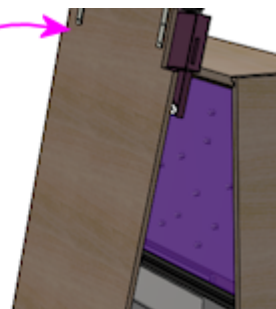
Pull out

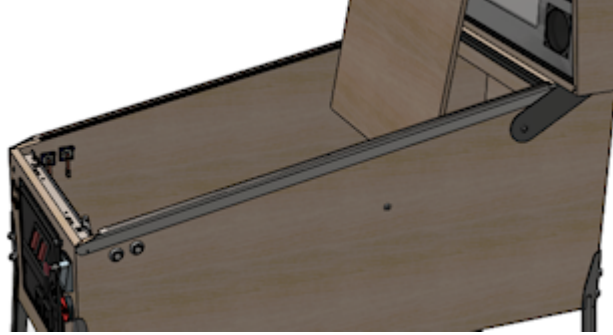


Lift the rest
of the way



Rest against
backbox



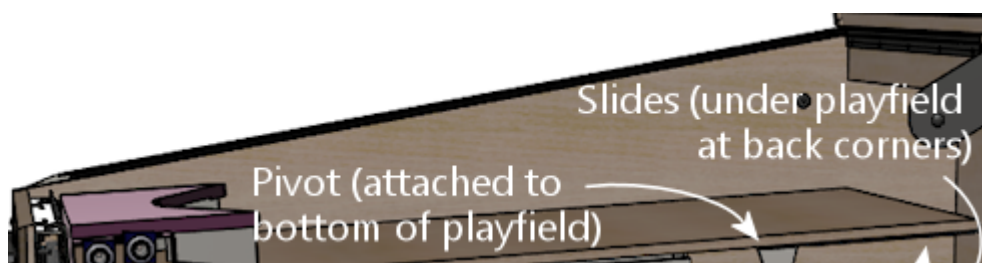


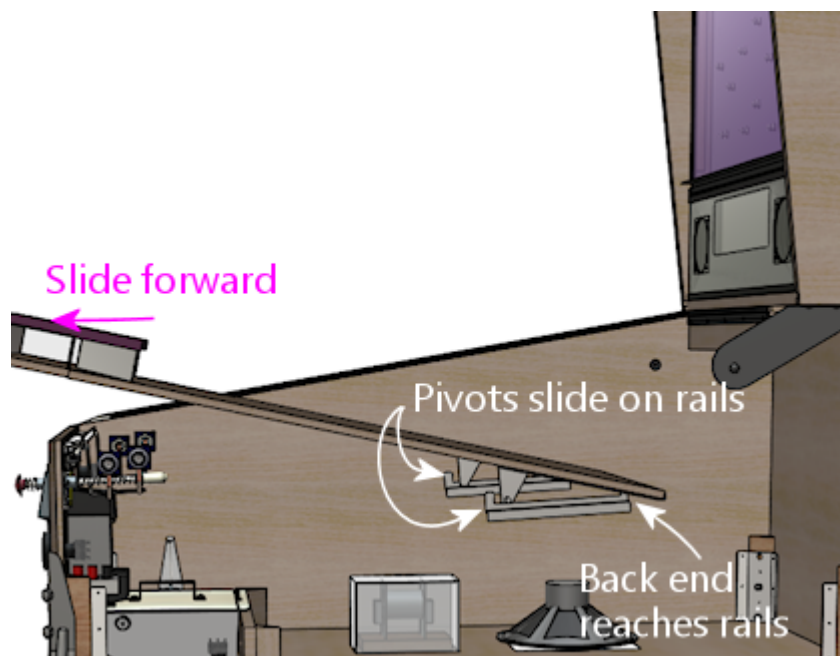
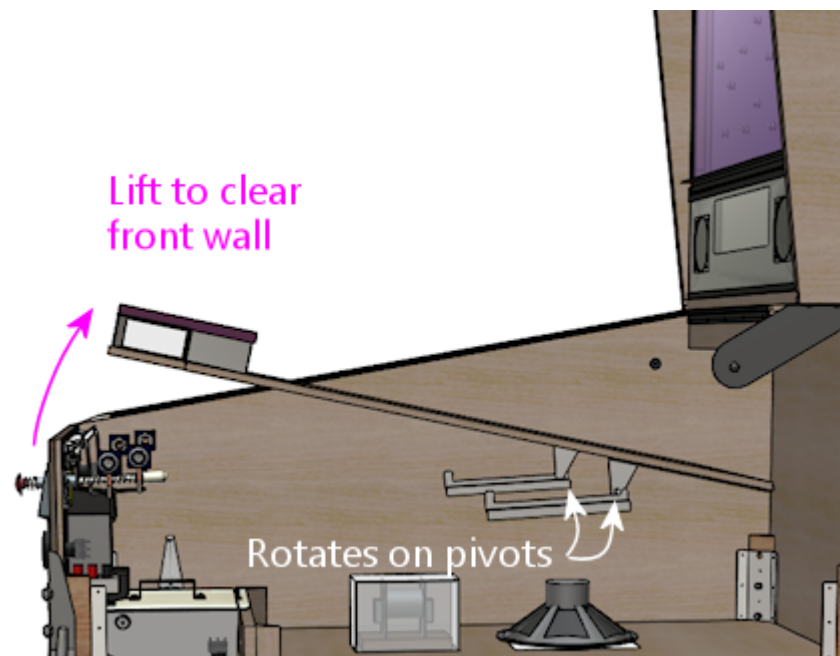
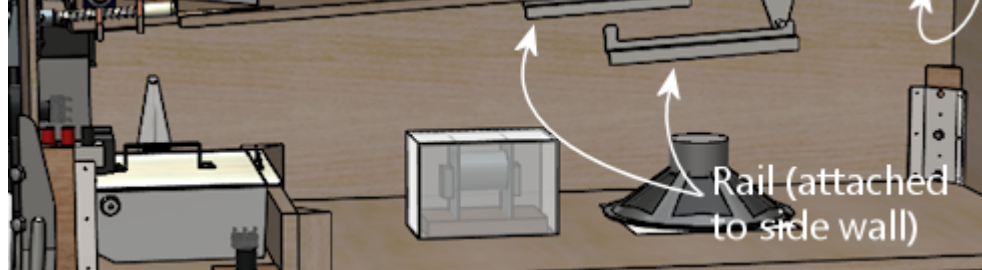
This approach might look attractive for a virtual cab, too, because it solves some of the geometry problems that we discussed earlier in "Determine the hinge position". But on closer inspection, I don't think it actually works that well for a virtual cab. The problem is that moving the pivot point forward like this ends up blocking access to a larger portion of the cab interior. That's fine on a real pinball machine, because the cab interior is mostly empty anyway - most of the parts you want to get to for service are on the underside of the playfield. But in a virtual cab, we tend to install lots of stuff in the cab, so the slide-and-pivot design works against us in that respect.

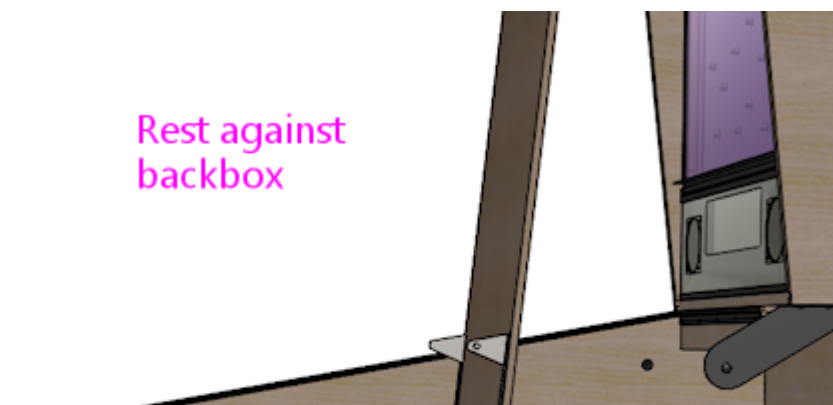
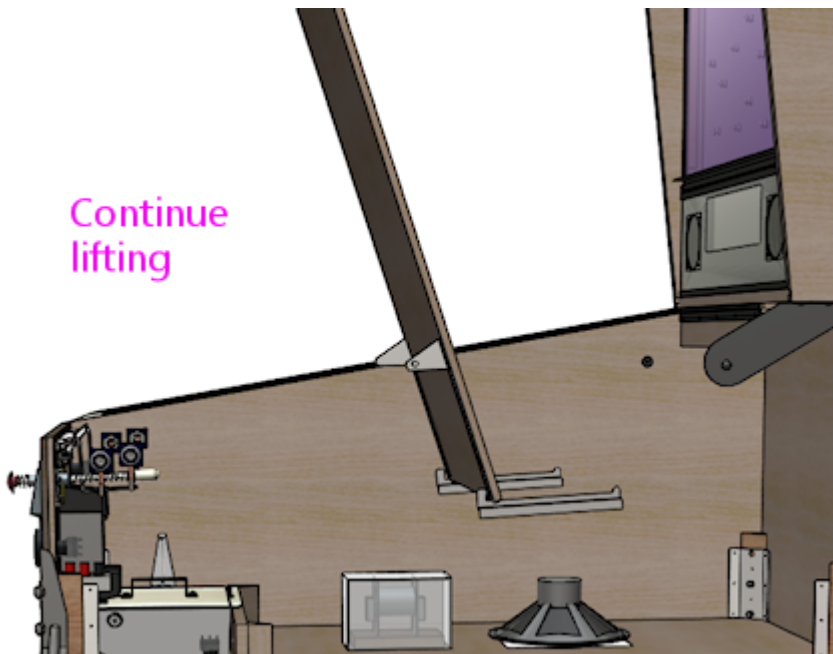
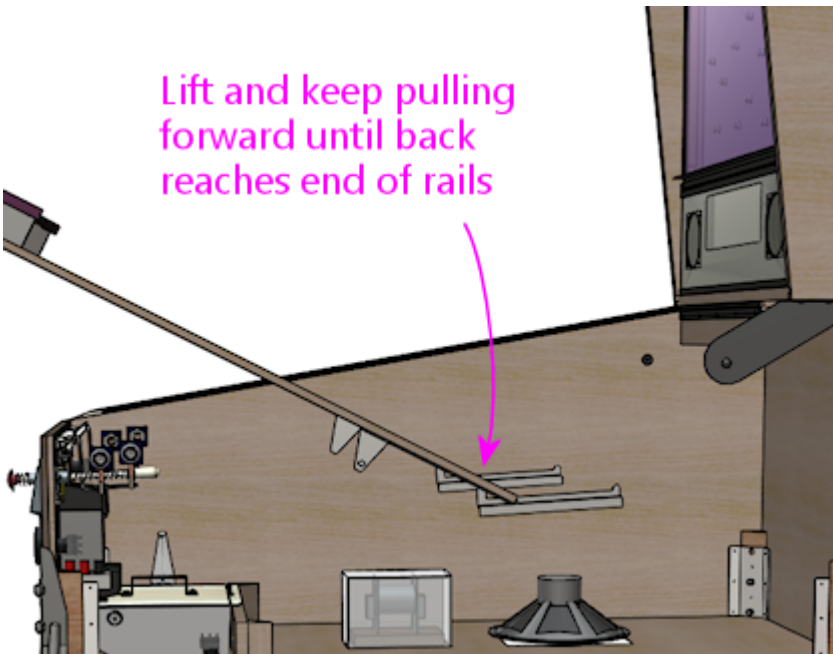
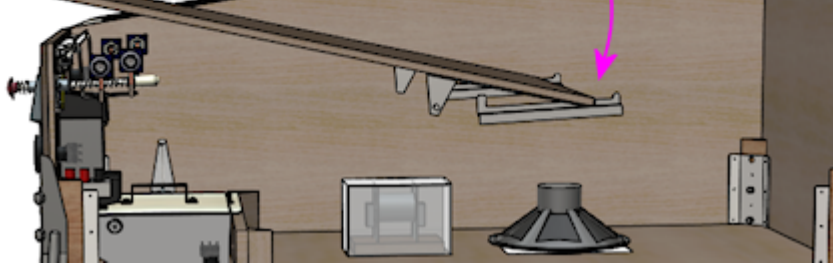
Given those geometry drawbacks, as well as the added complexity, I don't think most virtual cab people will want to implement a design like this. So I'm only going to offer an overview of how it works, rather than going into such great detail as I did with the fixed hinge system. I'd be happy to revisit the subject if there's enough interest, though, so let me know what you think.

In the Williams WPC machines, they implement the pull-out system with a rather complicated "slider bracket" mechanism, which uses spring-loaded levers and latches to provide the sliding capability and lock the pivot point in place at the forward and aft positions. The slider brackets are mounted to the bottom of the playfield, taking the place of the simple pivot brackets of the older design. As in the older design, the slider brackets rest on top of pivot nuts mounted to the side of the cabinet. The difference is that the slider brackets let you move the playfield back and forth across the pivot point, so that you can pull it out (as illustrated above) before tilting it up. The slider bracket system works nicely in the real machines, and I think it would be possible to adapt to a virtual cabinet, but the parts are expensive - about \$75 for a set of the slider brackets. If you're interested in investigating further, the Williams part numbers for the slider brackets are A-17749.1-1 (left side) and A-17749.1-2 (right side); those mate with the same pivot nuts (02-4244 or 02-4329) and 3/8"-16 x 1-3/4" carriage bolts as in the older design.

Stern's modern machines also have a pull-out system, but they use a completely different mechanism. Stern's system is simpler and cheaper, but some Stern owners say that it's a bit clunky to operate. Stern's design essentially inverts the Williams design: it uses pivot pins on the *bottom of the playfield* instead of on the side walls, and has rails mounted on the side walls that the pivot pins ride on. To slide the playfield forward, you slide it along the rails until it reaches stops at the front. The clunky part is that the Stern rails don't have the spring latches that lock things in place at the different positions in the Williams design; instead, they rely on gravity and little bumps in the rails. Here's an illustration of how the mechanism works:









As you can see in the illustrations (which I've tried to keep to scale), the bottom of the playfield ends up positioned much further forward in this design than in the fixed-hinge system. That's really the whole point, since that's how this system deals with the geometry problems of the fixed-hinge system, but it's a negative for a virtual cab because it only gives you access to the front half of the cab. You're likely to have things mounted further back than this.

If you do want to go with this system, I think you could easily adapt the step-by-step installation procedure outlined earlier for the fixed hinge system. The parts in the two systems are analogous, so you should be able to use the same techniques to measure and align everything. Here are the Stern parts you'd need:

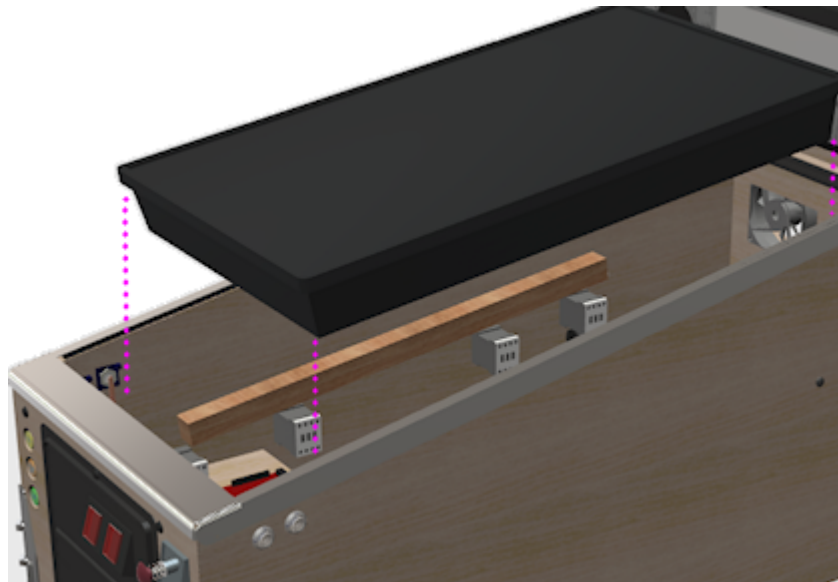
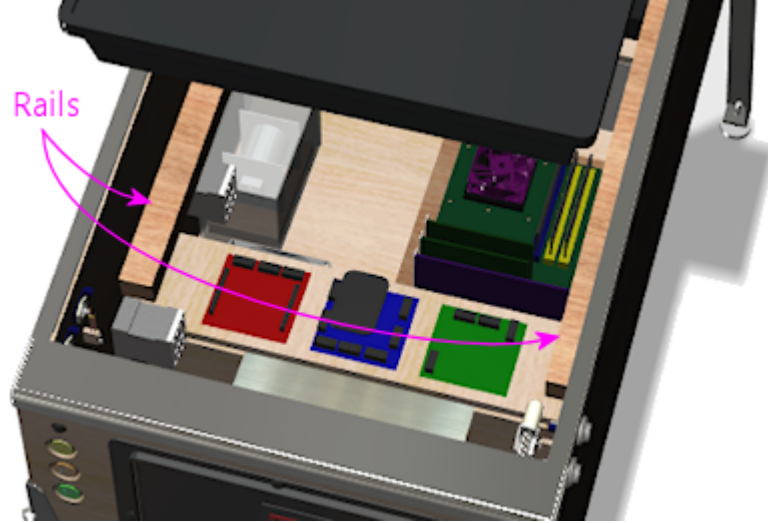
- Pivot brackets, 500-5329-03 (alternates: 500-5329-00, 500-5329-01, 500-5329-02) (quantity 2)
- Left side wall support rails, 535-5990-00
- Right side wall support rail, 535-5989-00
- Carriage bolts, #10-24 x 1-1/4", black (to attach the side support rails to the walls) (quantity 6)
- #10-24 hex nuts (quantity 6)
- #10 lock washers (quantity 6)
- Playfield slides (attach to the rear outside bottom edges of the playfield), 535-5988-01 (quantity 2)

Also see the EZ slide playfield support brackets at Back Alley Creations. That's an after-market replacement for the original Stern support rails that's supposed to offer smoother operation. (The Stern parts are all metal, without any wheels or bearings, so there's a lot of metal-on-metal scraping involved. The "EZ slide" replacements use a smooth plastic for the rails to make it less of a nails-on-chalkboard experience.)

Rail mounting

A simpler alternative to the tilt-up mounting is to rest the playfield on rail supports along the sides of the cabinet. This still lets you access the cabinet interior when needed, by removing the TV, although it's not as convenient as simply lifting the TV without removing it as you can do with the tilt-up design.





While this is a little simpler to build than the tilt-up mounting, I wouldn't use it myself. I consider it too inconvenient, since you'd have to entirely remove the TV (and unplug all of its cables) every time you wanted to get into the cabinet. That would turn minor work into a big hassle.

Some random thoughts if you use this type of design:

- For the supports, I'd only use rails along the sides of the cabinet, not across the width of the cabinet. Cross-bars make it more difficult to reach into the cabinet.
- Be sure your TV is adequately supported underneath. Some TVs might not be strong enough to be suspended from the sides only. You might still want to mount the TV to a plywood base via its VESA anchors, as in the tilt-up mounting design above, then rest the plywood base on the rails rather than placing the TV directly on the rails.
- A base would also allow you to build the flasher panel and/or apron area into the TV assembly, as in the tilt-up design. This lets you remove the whole thing as a unit when you need to access the cabinet interior.

Routed slot mounting

If you want to use a TV that's slightly wider than the interior width of your cabinet, you can route grooves in the side walls that the TV fits into.



This is similar to the "rail" design above, in that the TV is supported from the sides. As with the rail design, you should make sure that your TV is strong enough to be suspended this way, and if not, add some kind of base underneath to support it.

The main attraction of this approach is that it can produce a "wall-to-wall" video screen effect, by hiding the TV's bezels in the grooves. I can see the appeal, but it has too many functional tradeoffs for me. In particular, with a routed slot mounting, you'd only be able to get the TV in and out by sliding it through the front wall. In other words, you'd have to *remove the front wall* to get the TV out, and that's impossible with a conventional cabinet design, since the front wall is permanently attached. To me, it's a must that you be able to access the interior of the cabinet at any time, so I wouldn't use this approach unless I could find a way to make the TV easily removable without having to disassemble the whole cabinet.

Permanent installation

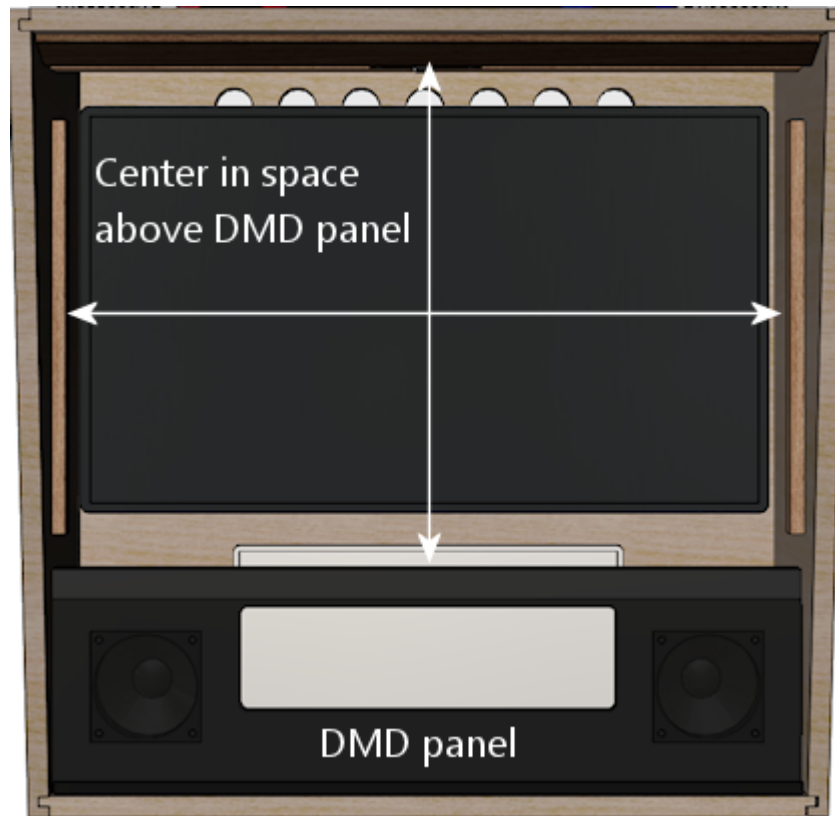
Some people install their TVs permanently, fastening them directly to the cabinet with screws, nails, or glue, with no provision for tilting up the TV or otherwise moving or removing it. You can't beat the simplicity of this approach, but I'd personally rule it out for any cab project of my own, since it would make maintenance and repair so difficult.

30. Backbox TV Mounting

Continuing the theme of installing our TV screens, we turn now to installing the backbox TV.

Positioning the TV and light-sealing

I recommend positioning the TV so that it's centered in the space above the speaker/DMD panel.



If you're using a translite cover, I'd try to position the front surface of the display so that it's flush with the translite, or as close to flush as you can make it. That will make it look very much like the artwork is being displayed on the back of the glass/acrylic, which looks just like the real thing.

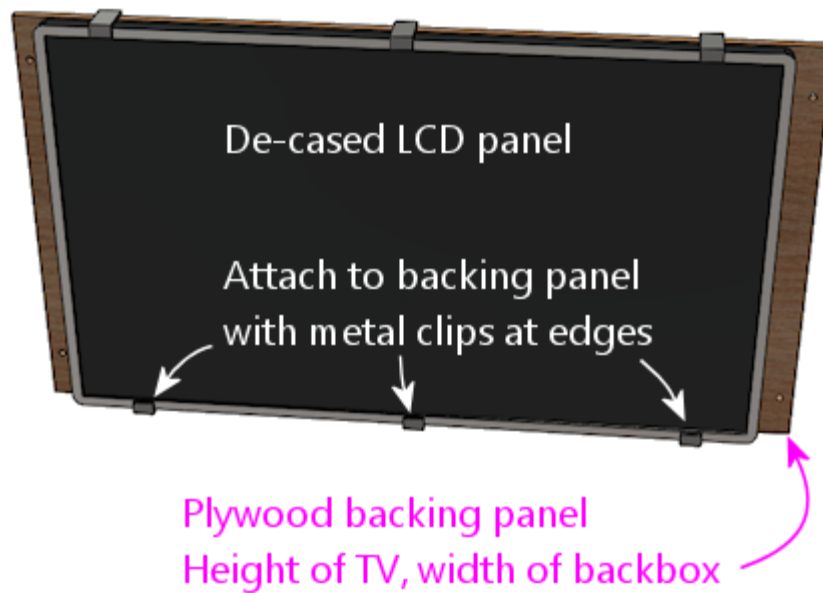
On my machine, I put strips of black felt around the perimeter of the display bezel (fastened with double-sided tape). This lets the translite be right up against the monitor face without too much risk of scratching it, and it also forms a nice light seal around the perimeter.

The Jersey Jack Pinball approach

The WPC-era machines didn't have anything resembling a full-sized TV in the backbox, so there's nothing we can directly adapt from their design or hardware to mount the TV. But if we widen our scope beyond the post-WPC era, we can find one example from the real pinball world that's very similar. Jersey Jack Pinball, maker of *Wizard of Oz* (2013), *The Hobbit* (2016), and a few other titles, uses a 27" LCD panel in the backbox in place of the WPC-era DMD. They even install a translite (with a cutout for the TV) in front of the panel, so it's almost exactly like the typical virtual cab setup. The only difference is that they place the panel at the bottom of the backbox, whereas we typically place it near the top, so that we can also put a

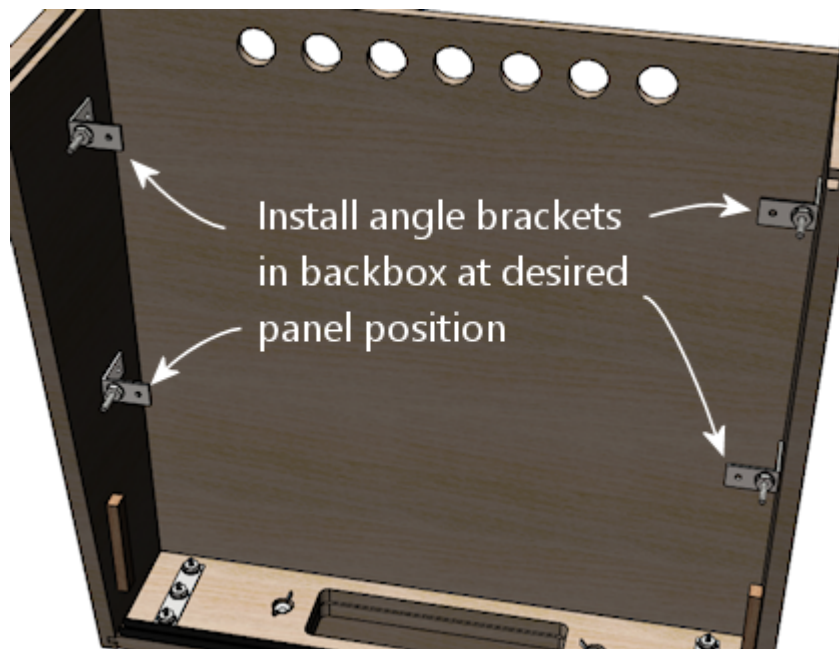
traditional DMD-sized display in the speaker panel area. But that difference in placement doesn't affect the mechanics of the mounting.

Their approach to mounting the panel is straightforward. They start with an uncased OEM display panel, and mount it to an MDF backing board with metal clips around the edges. For a virtual cab adaptation, I'd use plywood instead, since it's stronger and lighter.



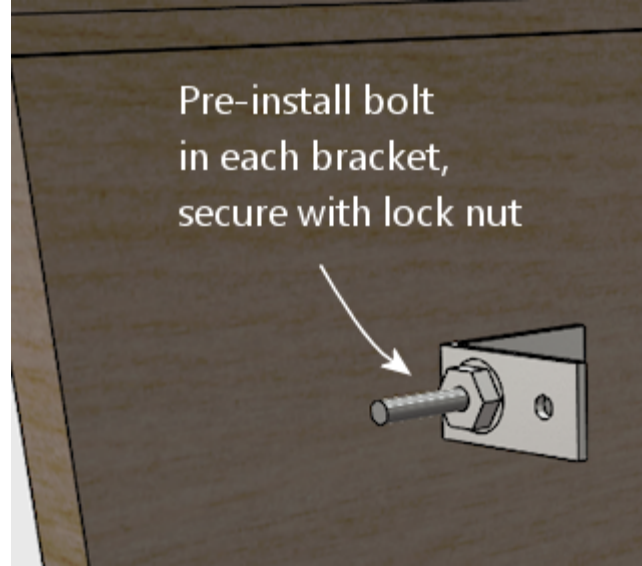
The backing panel is cut to the height of the panel, so that you can easily fasten it with clips as shown. The width is the same as the inside width of the backbox.

In the backbox, they install metal "L" brackets along the side walls.

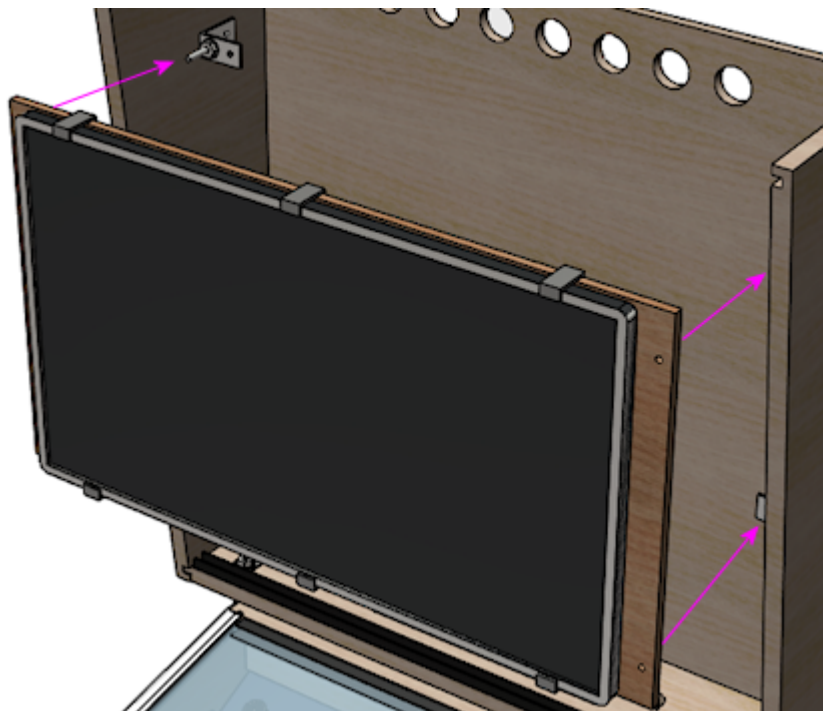


These can simply be screwed into the backbox walls with wood screws. The overall weight of an uncased panel plus plywood backing board should be under 10 pounds, so each corner clip only has to support a couple of pounds. So we don't need super-strong hardware here.

To facilitate installing the panel, pre-install a bolt in each bracket, with the end pointing forward as shown below. Secure each bolt with a lock nut.



Drill holes in the plywood backing to match the positions of the pre-installed bolts. Installing the TV is now just a matter of fitting the backing panel onto the bracket bolts. Secure it in place with another set of nuts on the outside.



Some notes on this approach:

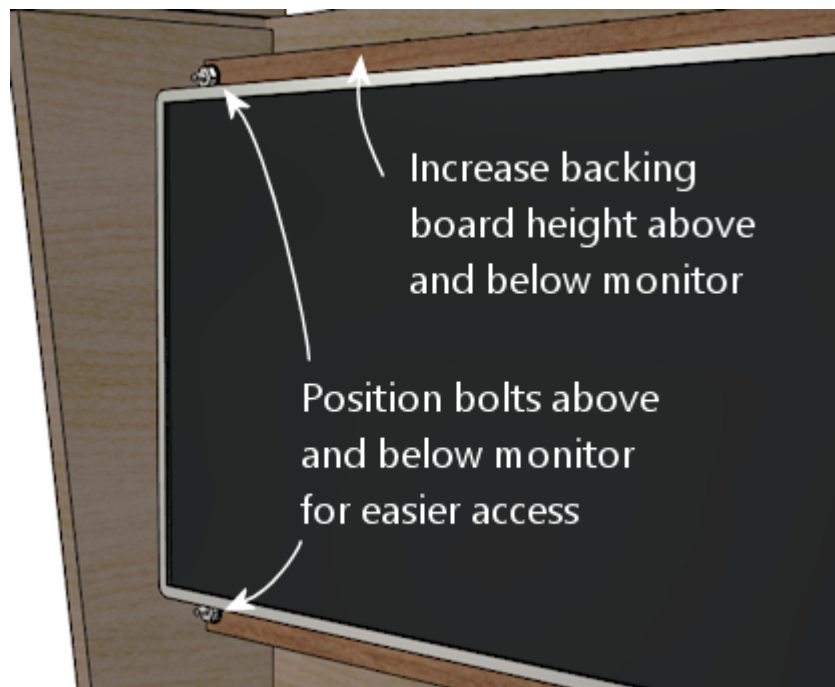
- When constructing the backbox, **don't** install the upper (15") translite guides. Those would get in the way of inserting and removing the TV panel. See "Translite/DMD guides" in Chapter 21, Cabinet Body.
- You might replace the outer nuts with wing nuts, so that the TV can be installed and removed without tools. Better still would be some kind of quick-release latches - if you have any concrete ideas along those lines, let me know and I'd be happy to add pointers here.

Adapting to a TV still in its case

This design works well with a bare, uncased LCD panel, but it requires some changes if you're leaving the TV in its plastic case.

First, use the TV's VESA mounting holes on the back to attach it to the plywood base, instead of the edge clips. You just need four M4 screws long enough to go through the plywood (probably 20mm to 30mm).

Second, with the added depth of a full case, it might be too cumbersome to reach behind the monitor on the sides to fasten and unfasten the nuts that hold it in place. So I'd move those so that they're above and below the TV, where you'll have a little more room to work.



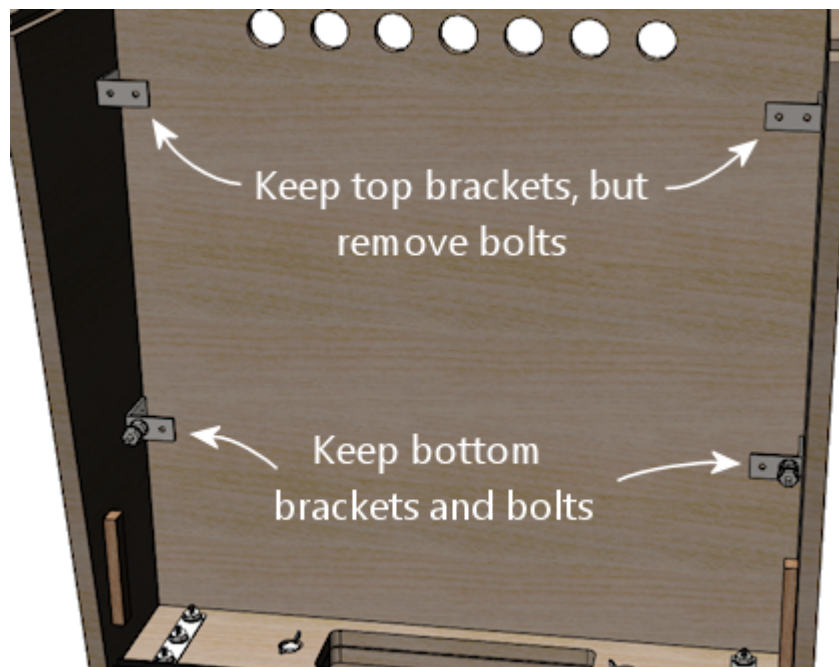
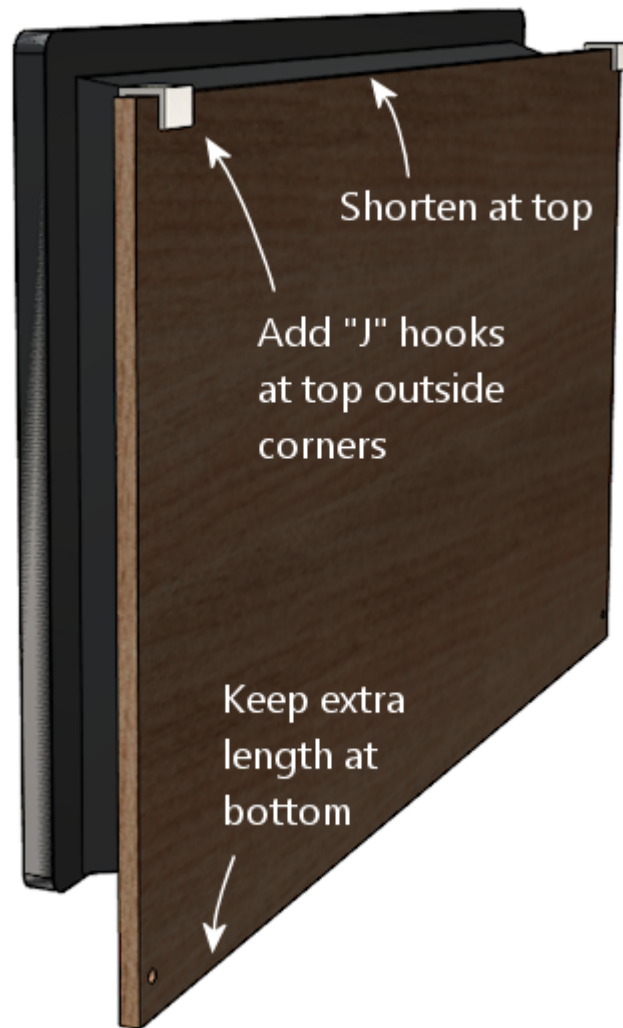
Further improving the design

Even with the bolts moved vertically above and below the TV in the revised plan above, I think it's going to be a little cumbersome to fasten those top bolts. The space above the monitor is quite tight with a 28" display.

So I think the next step in improving this plan would be to substitute some type of hanger hook at the top, such as a heavy-duty picture hanger or "J" bracket. Or perhaps better yet, the latch bracket used on the speaker/DMD panel (Williams/Bally 01-8535), which is designed for this exact function with the speaker panel.



The bottom bolts are easy enough to access that we can keep those.

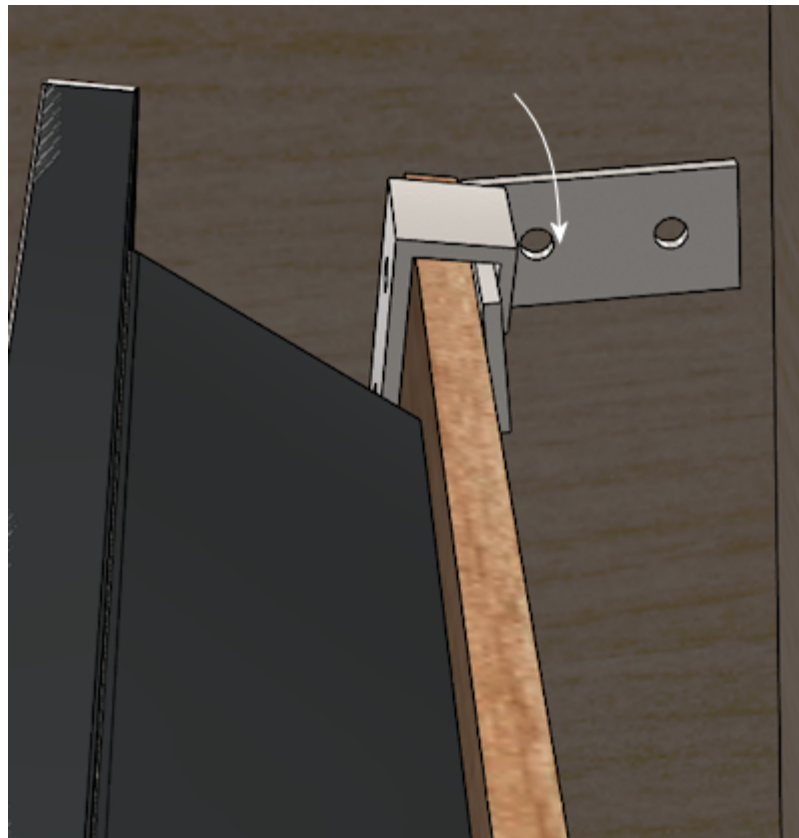


The new installation procedure would look like this:

- Tilt the monitor back slightly and fit into the hanger hooks at the top

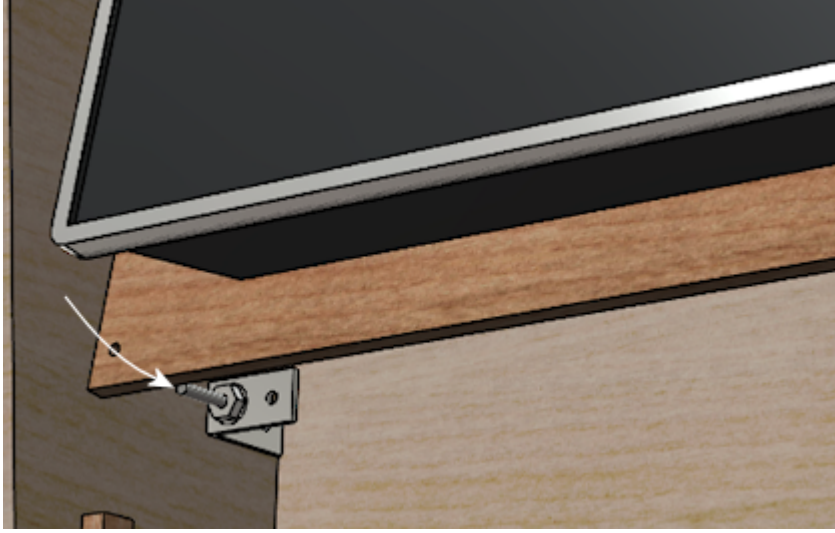


- Lower onto the hooks



- Tilt the bottom onto the bottom bolts





- Fasten the nuts on the bottom bolts

I used a mounting similar to this on my own cab, and it works pretty well. This should give you a little more than an inch of clearance behind the TV for installing a replay knocker and EM-style shell bells. You'll be able to access those devices for service if necessary, since the TV can be removed fairly easily.

Fixed mounting

Another, easier way to mount a TV in the backbox is to more or less build the backbox around the TV. You get the TV situated and bolted down while the backbox is still being assembled, and then you finish installing the remaining backbox walls.

There are different ways to go about this, but I've seen people use procedures something like this:

- Start by building the backbox *without* its back wall
- Cut a piece of plywood to the interior dimensions of the backbox
- Fasten the TV to the plywood (via its VESA mount, for example)
- Fasten the plywood to the side walls at the desired depth, such as via "L" brackets or corner braces
- Install the back wall of the backbox

If you've read much else in this guide, you can probably guess that I don't recommend any approach that would make it difficult to remove the TV later. If you have to wait to finish the backbox assembly until the TV is installed, you're going to have to reverse those last assembly steps if you ever want to remove the TV, and that could be difficult and destructive.

An argument could be made that it's okay that you can't remove the TV, because why would you ever need to? Modern consumer electronics tend to keep going for years and years without trouble. That's usually true, but not always, plus it ignores the possibility that you might need to access or remove the TV for some reason other than repair: upgrading it to a newer model, say, or changing which video plugs you're using.

There's also a potentially bigger problem: it'll be impossible to access anything installed *behind* the TV without taking the back off the backbox again. There's at least one part commonly mounted behind the TV that you might at some point need to repair: the replay knocker. This isn't a hypothetical risk, either. I've talked to a

couple of people on the forums who had to do major surgery - including taking out nails and glue - to replace a dead replay knocker. So I really strongly advise against any design that would seal up any important components behind barriers that you can't remove, such as a TV with a fixed mounting.

If you are using a fixed mounting, then, you might want to do one of the following:

- Don't install anything behind the TV in the first place, so you'll never need to get back there. Treat the space behind the TV as a forbidden zone that no parts can set foot in. This solves the access problem, but you give up some prime real estate in the deal.
- Make the back wall removable, or build a big door into it. This restores access, but at the cost of significantly weakening the backbox. The fixed back wall is a key structural component.

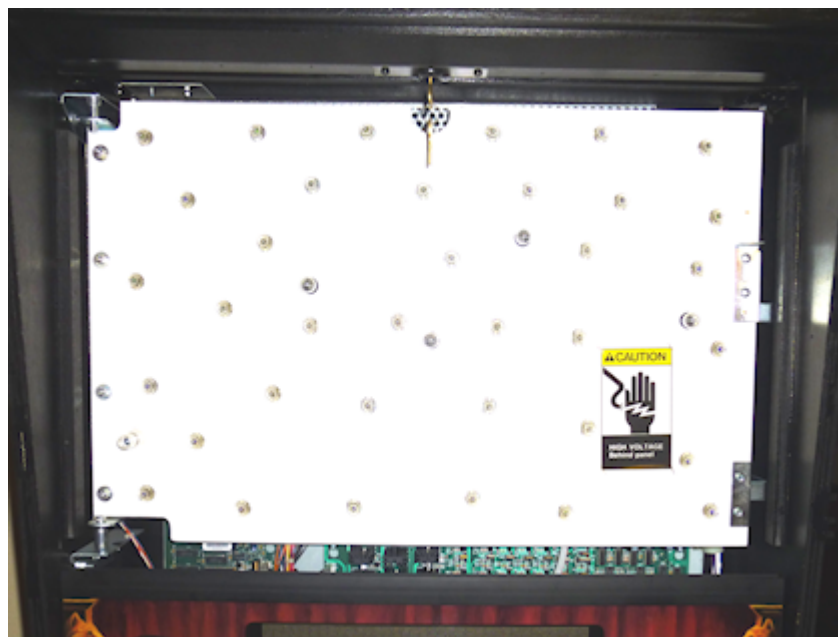
I don't like any of these trade-offs, so I just wouldn't use a fixed mounting like this.

Ideas from the real pinball world

What follows isn't a how-to plan, but just some food for thought. I'm hoping someone can take the ideas here at some point and turn them into a workable implementation plan. For now, though, these are just some ideas.

The WPC-era machines didn't have anything quite like a backbox TV that we can adapt in terms of mounting hardware, but they did have something analogous that at least provides an interesting idea for how a TV mounting might function.

The WPC machines have something called a "backbox insert", which is a sheet of plywood directly behind the backglass, holding the little light bulbs that illuminate the artwork. This is *exactly* where our backbox TV goes, so it's worth looking at how they installed this.



Backbox insert: a sheet of plywood installed just behind backglass. The little spots scattered around are the lamps that back-light the artwork.

On the WPC machines, all of the control electronics are mounted inside the backbox, so you obviously have to be able to move the insert out of the way for service work. The WPC design made this really easy. You don't have to disassemble anything or

even remove the insert. It's attached with hinges on the left side, so you just swing it out of the way like opening a door.

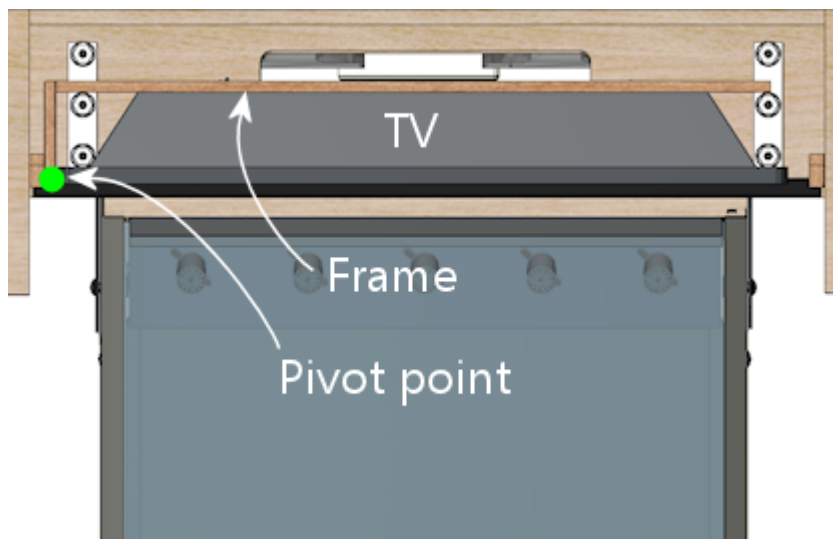


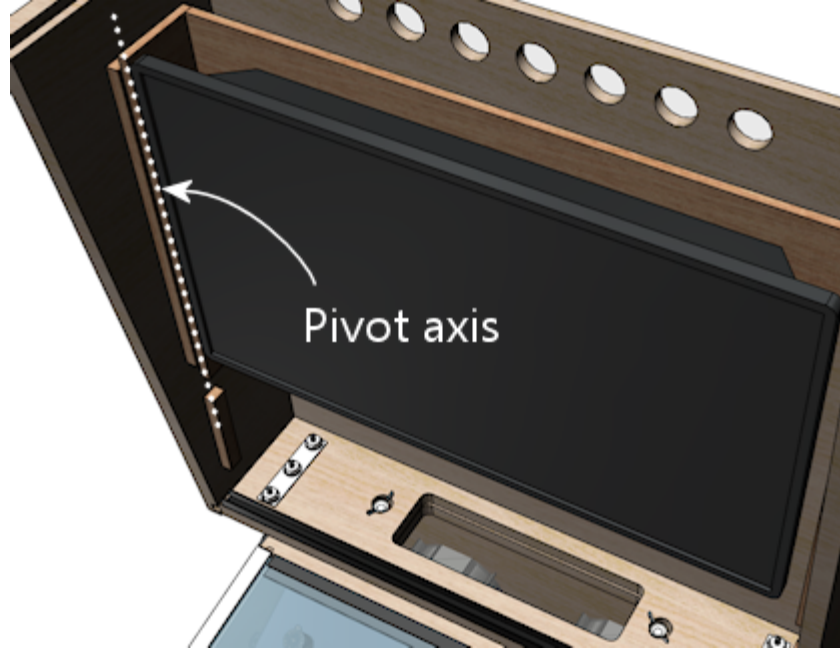
Backbox insert in open position. It's hinged on the left side so that it can swing open like a door, to provide access to the electronics installed behind it.

It would be great to be able to do the same thing with a TV - just swing it out of the way when necessary, without even having to unplug any video cables.

I'm afraid I don't have a How To plan to offer here to accomplish this, though. My first thought would be to try to adapt the original insert hinge hardware they used on the WPC machines, but those won't work for a TV; they have the wrong geometry for anything deeper than a sheet of plywood, and I don't think they'd be strong enough for a 10-15 pound TV on a long lever-arm like this.

My next thought would be to look for some generic hardware that could do the same thing, but I haven't found any. The sticking point is the depth of the TV. To make the geometry work, we need to place the pivot point at the front corner of the TV, and that means that the TV itself has to be mounted on an articulated platform. To use the TV's VESA mounting, this has to articulate by the depth of the TV. The apparatus would have to look like this:





I think this can be done in principle, but I don't have a concrete proposal for how to build it. I'm not sure how to make a TV platform in that shape that would be strong enough. I don't think plywood is strong enough, given that the full weight of the TV has to be supported at that back left corner (in the illustrations). It might also be challenging to find suitable hinges, although that's probably just a matter of some legwork on Amazon, as there are lots of pivot hinges available; there's probably something strong enough in the right size.

If you can come up with a mechanism that would accomplish this with generic hardware, please let me know! I'd be thrilled to be able to include it in this guide. But for now, I'm going to say that a hinged mechanism isn't practical for our purposes here.

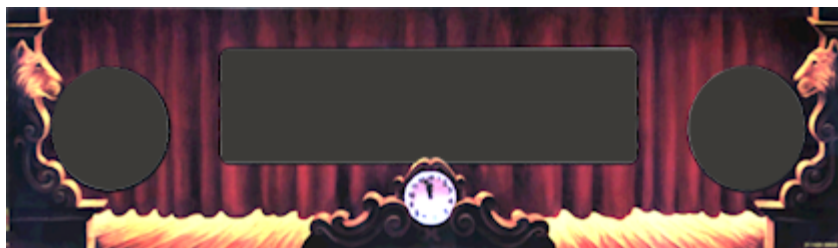
31. Speaker/DMD Panel

The speaker/DMD panel is a fairly complex piece of equipment, and new cab builders can find it challenging to assemble and install. This is one of those features of the real machines that's not well documented anywhere, and it's really not obvious looking at the parts how they're supposed to go together.

There are two rather different styles of speaker panels that were used in the Williams machines from the 1990s that most of us use as the model for our virtual cabs. Both types can be adapted to a virtual cab, so the first step in setting up your speaker panel is to decide which one to use. This chapter provides some details on the two types to help you decide. In the two following chapters, we'll provide detailed instructions for assembling and installing each type. They're different enough that we'll give each type its own chapter.

Original 1990s style

The first type of speaker panel that we'll look at is the "original" style that was used in Williams machines from the early 1990s to about 1995. This style uses an MDF panel with a front plastic facing silkscreened with game-specific graphics. For example, here's the panel from *Theatre of Magic* (Bally, 1995):



This design is something you can fabricate yourself, since the main elements are the MDF base (which you can make with a router) and the plastic facing (which you can have made by a laser-cutting service). You can also buy the pre-cut MDF panels and the plastic facing (in acrylic) from VirtuaPin.

The *original* original speaker panels that Williams used had asymmetrical speaker cutouts - a 5.25" cutout on the left and a 3" cutout on the right. This was because they used different speakers on the two sides: a large midrange driver on the left and a smaller tweeter on the right. This arrangement might seem strange in a modern context, where the whole point of a left/right speaker pair is stereo sound. But it makes more sense if you remember that all of the audio source material on these games is monophonic. They had no need for left/right channel separation. It's just a single-channel enclosure with a woofer and a tweeter.

For virtual cabs, everyone (almost everyone, at least) uses a matched stereo pair of full-range speakers instead of the midrange/tweeter combination that the real machines used. As a result, we don't want that bizarre arrangement with two different cutout sizes; we want the same cutouts left and right. So that's what we use in our plans presented later in this chapter: we give you the choice of two 4" diameter cutouts or two 5.25" inch cutouts. VirtuaPin's MDF panels are available with the same options. (VirtuaPin also sells the original asymmetric design for people replacing original equipment on real machines, but you probably don't want to use one of those for a virtual cab.)

Another small difference between the "original original" design and the reproduction design we use for virtual cabs is that the originals used a PETG plastic facing with the

graphics directly silkscreened onto the plastic, whereas a reproduction typically uses acrylic with printed decals. There's no practical difference in the results; the acrylic-with-decals approach is just a more DIY-friendly process.

WPC-95 panel style

Most of the Williams machines made in 1995 and later used a revamped design for the DMD panel, which we refer to as the WPC-95 style. In this design, they greatly simplified the manufacturing process by making the whole panel a single piece of molded plastic. They dispensed with the MDF base and the plastic facing, integrated the H-channel trim into the plastic form, and got rid of most of the extra hardware.

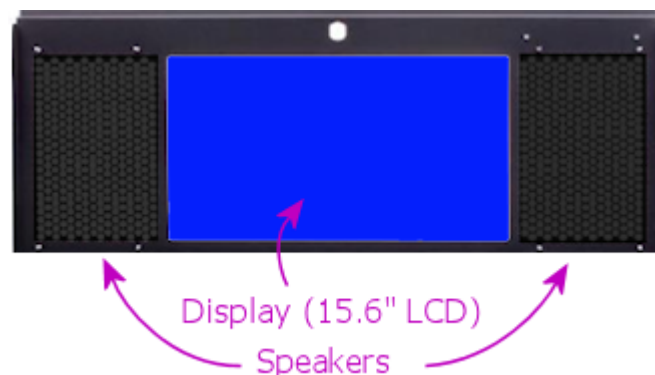
They also made the appearance more generic, to make the part interchangeable among games. The plastic facing with the custom graphics for each game is gone; instead, you just see the matte black face of the plastic panel. The only graphics on the production machines were a Williams or Bally logo.



Because of the single-piece molded plastic design, it's not practical to fabricate these as a DIYer. But you can buy new ones as replacement parts from any of the pinball suppliers, for about \$100.

Stern Spike 2 style

Many of the newer Stern games, made from about 2016 onwards, feature a 15.6" LCD video display in place of the 1990s dot matrix display. Like the WPC-95 panels, the Spike 2 panels omit any game-specific graphics.



Virtual cab builders have long used LCD panels of this same size to emulate the older DMD devices. 15.6" is a common panel size that's widely available, and it just happens to be almost exactly the same width as the traditional 128x32-dot plasma DMDs used in the 1990s machines, so it's a near-perfect fit as a DMD replacement. The only incongruity is that a 15.6" laptop panel is a lot taller than the original plasma DMDs, but that's an easy problem to solve: just stick the LCD panel behind a regular speaker panel with a standard DMD-sized cutout, and the panel hides the excess height. Some of the screen area goes unused, since it's hidden behind the

panel, but the player doesn't care.

The Stern Spike games take this same setup and open up the cutout vertically to show the full height of the panel. This lets the Stern game designers use the full 16:9 display area rather than confining the graphics to the smaller (shorter) DMD area.

The Spike 2 speaker panel would work well in a virtual cab if you like the idea of exposing the entire video display, rather than simulating the older DMD look. I don't have enough information on these panels to provide fabrication instructions, but there's little reason to build your own anyway, since the original factory part is readily available. Here are the Stern part number references:

- Spike 2 display/speaker panel, 515-9842-00
- Speaker panel spacer, 545-9877-01
- Speaker panel hinge, 515-9845-00
- 4" speaker, 031-5004-02

Note that the Stern panel doesn't include a mounting bracket for the LCD panel, since the bracket has to be customized to fit the specific panel you choose. You'll probably have to improvise your own bracket.

This panel style might be interesting as future-proofing. At the moment, most Visual Pinball simulations won't take advantage of the extra display height compared to the WPC DMD style, since their score graphics are exact reproductions of the original DMD layout, leaving most of the top and bottom of the screen blank. As time goes on, though, it's possible that newer games will take more advantage of larger displays. I imagine that the newer Stern games will eventually make their way into Visual Pinball simulations, at which point the original full-screen graphics would take full advantage of your larger display area.

Recommendations

My personal favorite style is the "original" 1990s style (the type with printed artwork on the plastic facing). I like the ability to customize the artwork to complement the overall cabinet theme, and the structure (MDF base and acrylic facing) is something that a DIYer can fabricate. What's more, the old design is flexible about what type of speakers you use, since you can cut the speaker openings to any size you desire (if you're fabricating it yourself), and you can drill additional holes in the MDF as needed for fasteners, in case the speakers don't fit the exact layout of the original design.

Some people prefer the WPC-95 style (the one-piece molded plastic design) because it looks more modern to their eyes, being the style used on the last generation of machines that Williams made. Some people also prefer it because you can buy the factory part rather than having to build something yourself. I think the plain black styling is a bit boring, but that's arguably a big plus for a virtual cab, where you want the cab to be a chameleon.

The Stern Spike 2 style is gaining popularity among virtual cab builders because of its larger display area, and because it looks the most modern, being the format that the newest Stern machines use. I personally have a nostalgic preference for the early 1990s look, as most of my all-time favorite titles are from that era, but that's an aesthetic preference. In terms of function, the Spike 2 format's larger display area is clearly more versatile, and will leave your system better prepared for virtual versions of the newer Stern games with full-screen video.

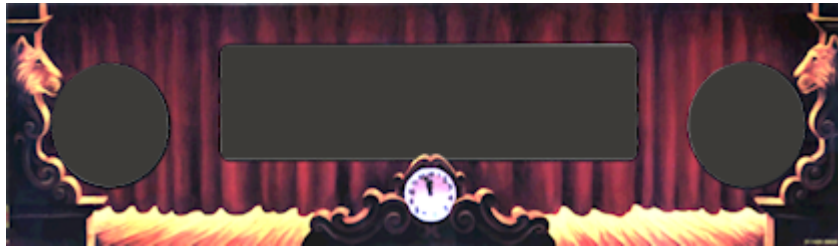
Assembly and installation

The details of the two panel types are very different, so we'll break them out into separate chapters:

- Chapter 32, Original WPC Speaker Panel
- Chapter 33, WPC-95 Speaker Panel

32. Original WPC Speaker Panel

This chapter goes into the details of the "original" WPC style of speaker/DMD panel, based on the ones used in the Williams and Bally machines of the early 1990s. These panels feature flat plastic front facings with silk-screened graphics, customized for each game. As an example, here's the speaker panel from *Theatre of Magic* (Bally, 1995):



Panels for other games from this era have the same overall shape, with the same cutouts for the speakers and display, but each game has its own unique graphics that harmonize with the backglass artwork (which is, of course, positioned immediately above this panel).

Around 1995, Williams switched to a different style of panel, which dispensed with the artwork to make them interchangeable among games, and which was constructed as a single molded piece of plastic to reduce assembly cost. See Chapter 31, Speaker/DMD Panel for more about the two main types of panel, and see Chapter 33, WPC-95 Speaker Panel for details on the WPC-95 type in particular.

Buying or building

Fabricating your own speaker panel from scratch is a fair amount of work, because it requires a lot of precise cuts with a router. If you do want to do it yourself, we'll give you complete plans later in this chapter.

If you want to buy a fully pre-fabricated panel, you can get a "blank" panel (without any game-specific graphics) from VirtuaPin. These come fully assembled with all of the required parts (MDF base, clear acrylic facing, H-channel, speaker grills, and fasteners). You just need to add a decal for graphics. The price is about \$130.

Notably, none of the pinball supply companies currently sell the MDF base panels. VirtuaPin seems to be the only place you can get those. The pinball suppliers do sell all of the miscellaneous hardware, such as the latch brackets and H channels, but not the MDF base. The Marco Specialties catalog lists a large number of game-specific "speaker panels", but those are just the silk-screened plastic face plates - they don't include the MDF base. The same is true for almost every "speaker panel" listing you'll see on eBay. If you're looking for full ready-to-use panel, check carefully before buying to make sure you're getting the complete panel and not just the facing.

There are also two options for partially building the panel, with some help with the trickiest wood-working portion:

- VirtuaPin sells the bare MDF base, with all cutting and routing done, but without any hardware. That runs about \$30. This takes care of what I consider the hardest parts of making the MDF panel, which are cutting out the openings for the display and speakers. Those have to be precisely measured and neatly cut for the result to look professional. The rest of the work with this option is just installing a bunch of T-nuts and other hardware, which is comparatively

easy.

- You can have a laser-cutting or CNC service cut the panel outline, the speaker and display cutouts, and the drilled holes, all from a computer template, and then do the routing and assembly by hand. The price varies a lot by vendor, but you should be able to have this done for about \$30 including materials.

Installing the panel

Let's start with how you install the speaker panel in the backbox. This is one of the big mysteries for first-time cab builders, so I think it'll help with the rest of your planning to see how the panel fits into the backbox, and how you install it and remove it.

First, there's one set of parts that we have to install on the speaker panel. Specifically, we have to install the latch brackets that hold the panel in place. (If you already have a fully assembled panel with the latch brackets installed, you can skip this step.) These are Williams/Bally part number 01-8535, and they look like this:



They install on the back of the panel at the top corners:



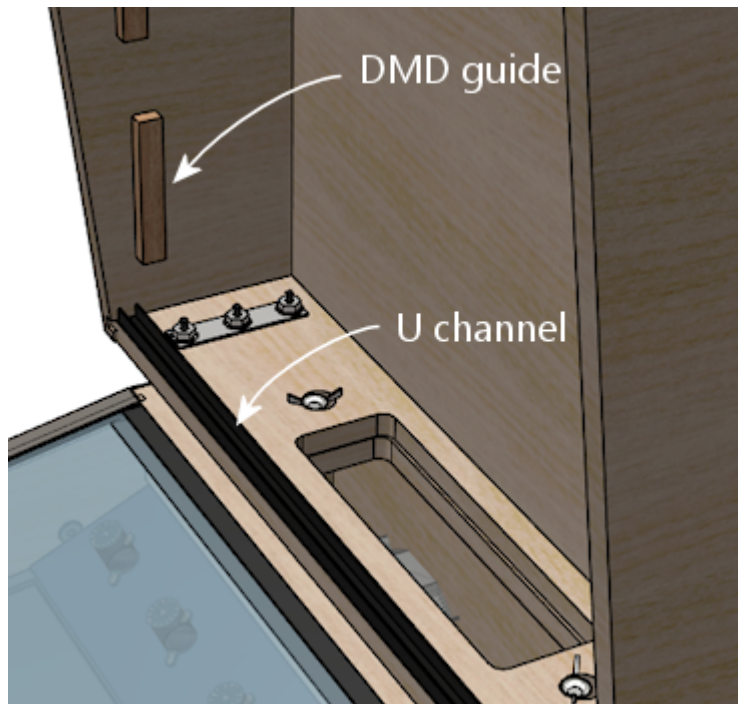
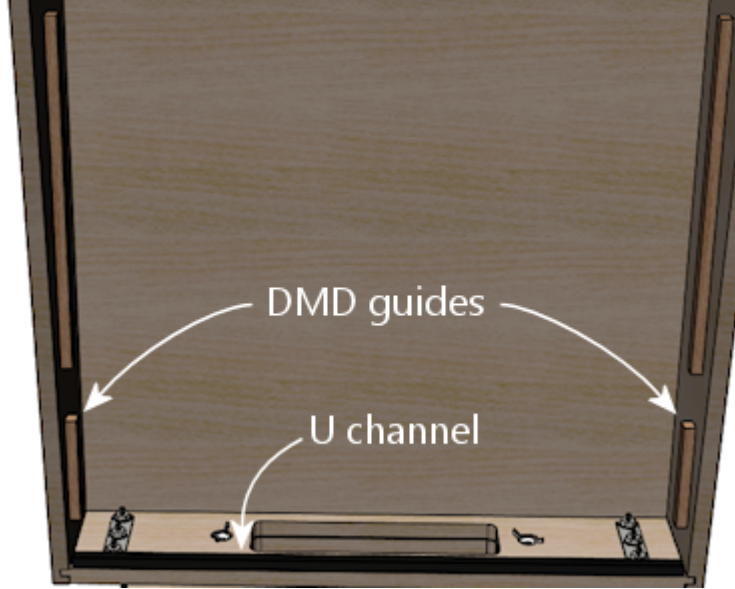
If you bought a pre-cut speaker panel from one of the pinball vendors, it should already have two holes drilled for each bracket, and a T-nut (#8-32) should be installed behind each hole, so all you have to do is line up the brackets and screw them in. Use #8-32 x 3/8" machine screws, of the "countersunk" type illustrated below. If you're fabricating your own panel, see our plans later in this chapter for the drilling locations.



Fasten the latch brackets with #8-32 x 3/8" countersunk machine screws

Next, there's some prep work in the backbox. The DMD panel is held in place by a couple of wood rails attached to the walls (which we call the "guides"), and a "U" channel at the bottom of the backbox.

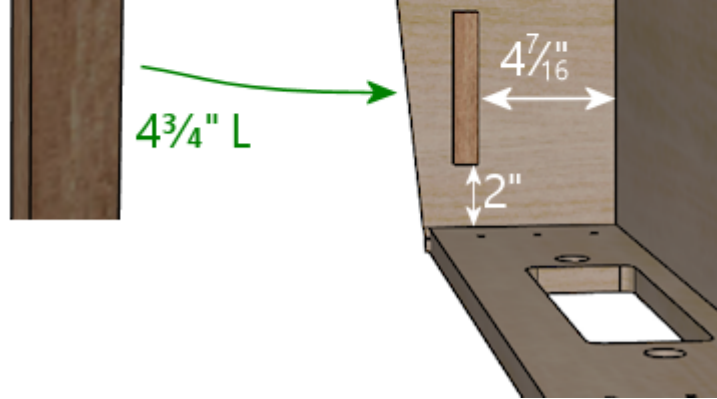




If you followed our WPC cabinet plans to build the backbox, you should already have installed the guide rails. If not, you should go back and put those in place now. The detailed plans are under "Translite/DMD guides" in Chapter 21, Cabinet Body. (Note that speaker/DMD panel only requires the lower set of guide rails described there. The upper set is for the translite, and you might not need or want those, depending on how you're installing your main backbox TV.)

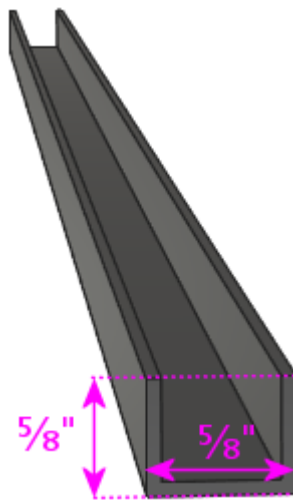
To save you a little time, here's a quick summary of the plan:





Our WPC cabinet plans didn't say anything about that "U" channel, though. We intentionally saved that for now, because it's easiest to get it aligned properly using the assembled DMD panel as a guide.

The "U" channel is also known as the Lower Speaker Panel Bracket, Williams/Bally part number 01-8569-1. The original part is available at PlanetaryPinball.com (none of the other pinball vendors seem to carry it), and you can also find upgraded replacements with custom finishes from "mods" vendors. You can substitute a generic $\frac{5}{8}$ " x $\frac{5}{8}$ " **U channel** from a hardware store, cut to the required length to fit your backbox width ($27\frac{1}{8}$ " for the standard WPC backbox).



This type of U channel is available in aluminum from Home Depot and other hardware stores. Note that it's only sold in fixed lengths (like 4' or 8'), so you'll have to cut it to the right length yourself. But that's easy to do with a hacksaw; the metal is pretty thin.

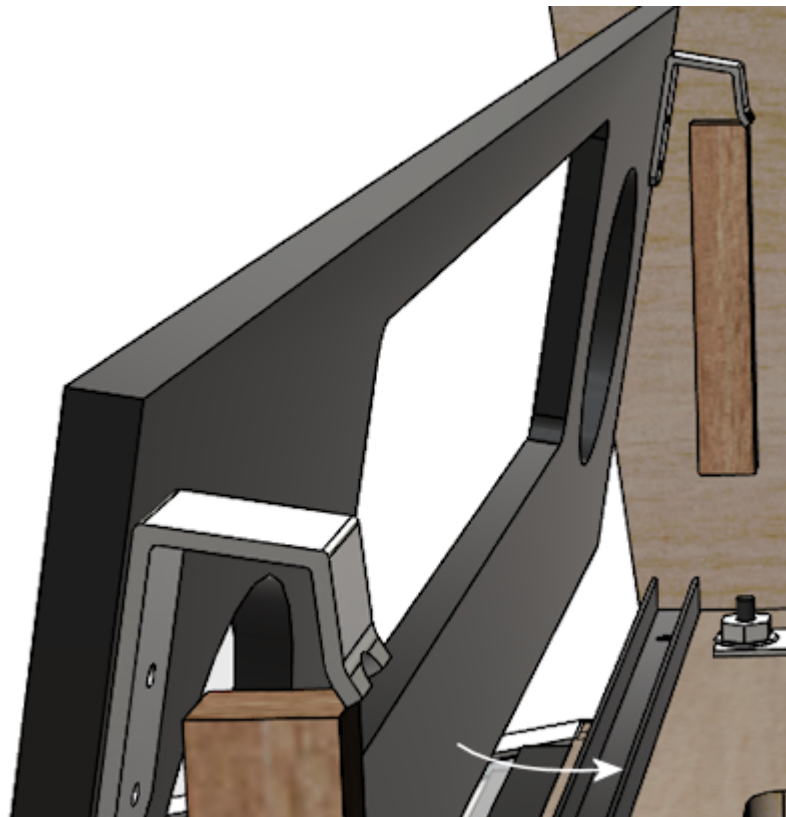
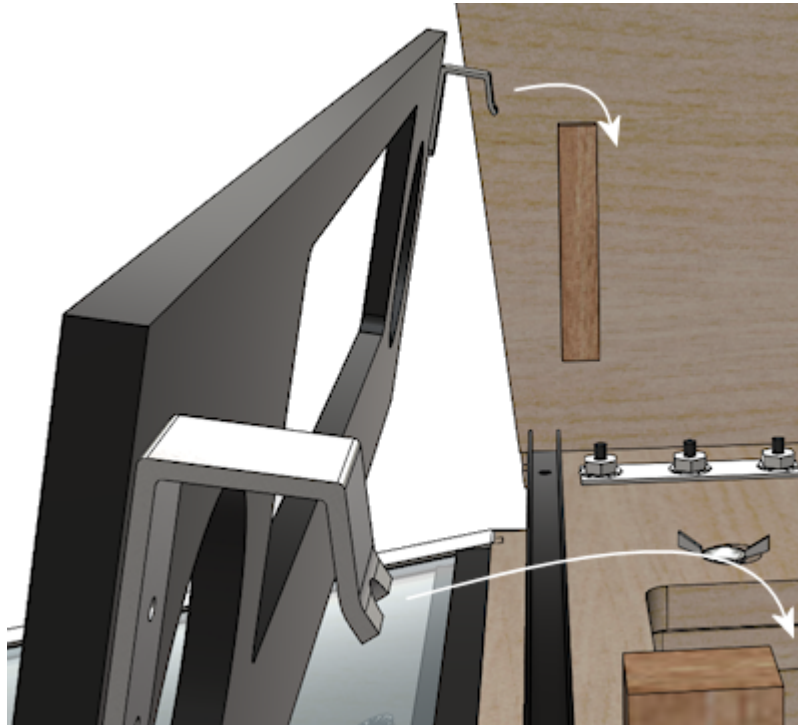
To prepare it for mounting screws, drill three holes in the bottom of the channel as illustrated below. I'd use $\frac{3}{4}$ " #6 wood screws for this, so drill to fit those. The exact locations aren't important; just eyeball them, one hole near each end and one near the center.



Paint it as desired before installing. In the original pinball machines, it was always painted to match the backbox interior color (usually black). But it is a visible piece of trim, so some people use a metallic finish to match the side rails and lockbar, especially if they're using custom finishes for those.

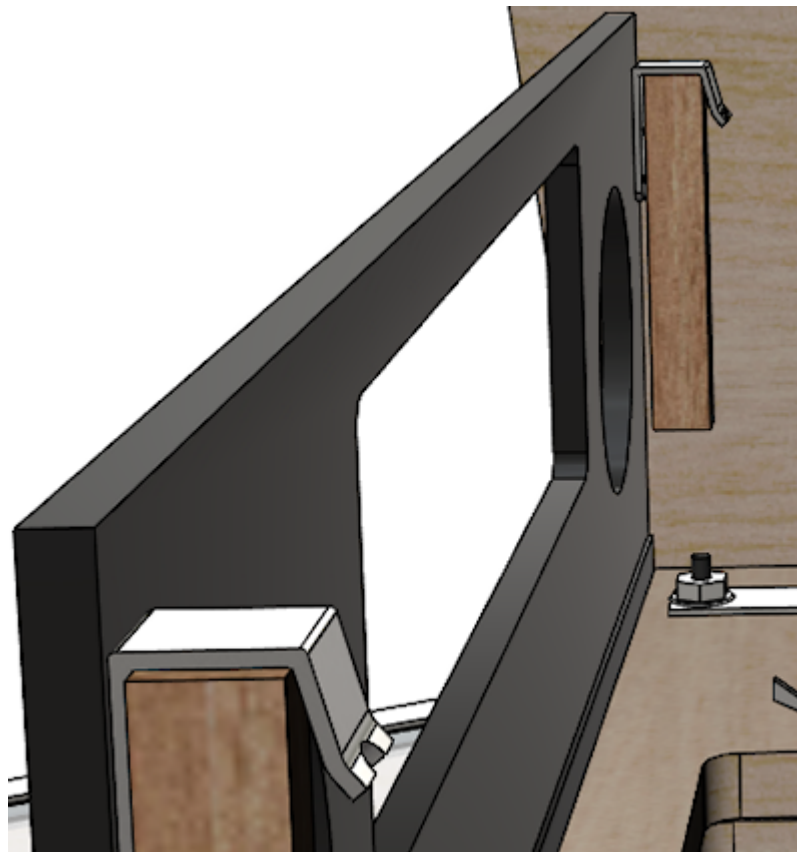
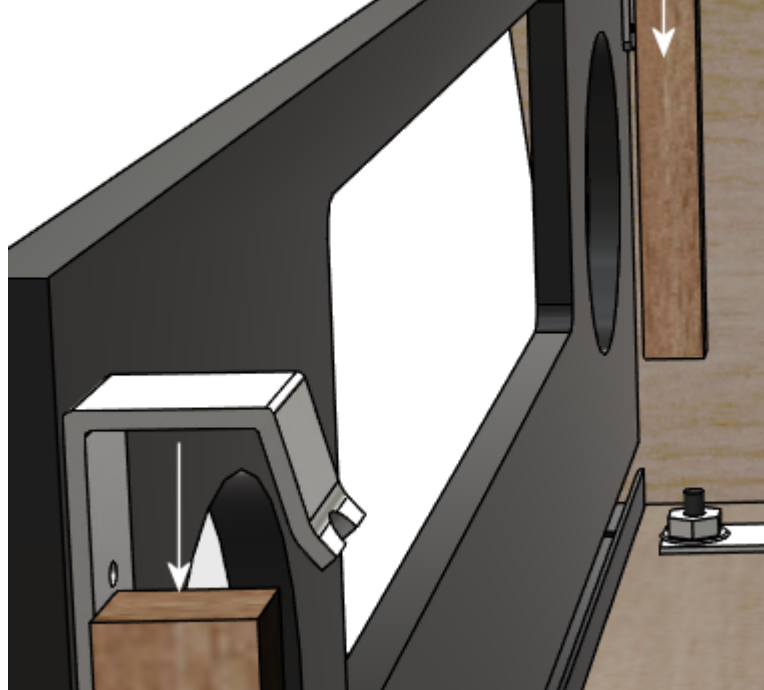
To install it, use the speaker panel as a template to figure the alignment:

- Put the U-channel on the floor of the backbox at about the front edge; this is the tentative position where it'll go
- Tilt back the speaker panel slightly, and fit the hooks on the back of the panel over the guide rails



- Lower it into position on the rails and into the U channel (adjusting the U channel position as needed), until the panel is resting in the channel



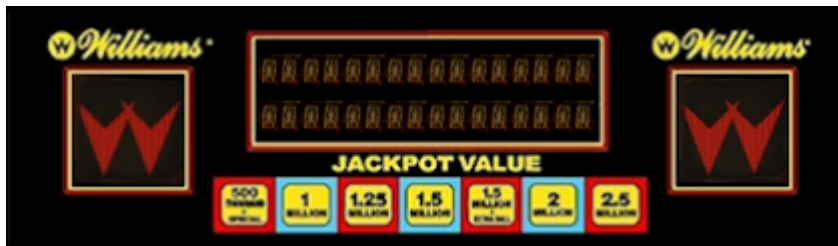


- Mark the position of the U channel
- Remove the speaker panel
- Make sure the U channel is aligned with the markings you just made, and secure it using wood screws

Any time you need to remove the DMD panel, simply lift it out of the U channel, lift the latch brackets clear of the guide rails, and it's free. To put it back, repeat the procedure above: fit the latch brackets over the rails and lower the bottom into the U-channel until it's firmly seated.

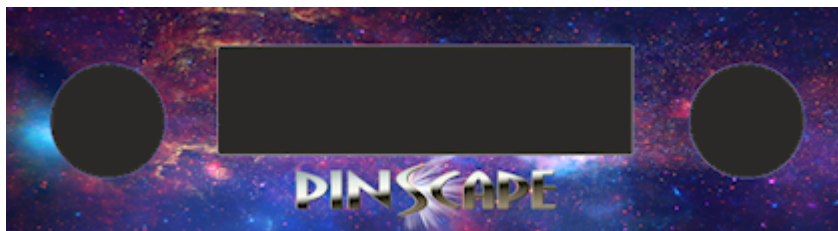
Designing graphics for the panel facing

The speaker panels on the real machines from the early 1980s to mid 1990s were printed with custom game-specific graphics. This was sometimes an extension of the backglass art, such as in *Theatre of Magic*, and sometimes included extra display features, such as *The Addams Family*'s ☆THING☆ lights or *Earthshaker*'s jackpot value display.



Speaker panel graphics examples: Theatre of Magic (Bally, 1995); The Addams Family (Midway, 1992); Earthshaker (Williams, 1989)

For a virtual cab, this is another area where you can create your own custom graphics. Here's my speaker panel, for example:



Printing the decal

There's a slight complication in using a decal for the speaker panel graphics: the cutout in the middle for the display.

With all of the other decals in your cab, holes in the middle are no problem. You can just cut them out with an X-acto knife after installing the decal, by running the knife around the edge of the hole. That technique works for the flipper button holes, coin door cutout, etc, but it won't work here! The problem is that we're going to affix these decals to the clear acrylic facing, and *there's no DMD hole to trace* in the acrylic facing. In the original design that Williams used, and in both our plans and VirtuaPin's products, the acrylic facing is continuous across the DMD area with no cutout. The speaker openings *are* cut out, by necessity, but not the DMD area. This was intentional in the Williams machines, probably mostly to protect the display

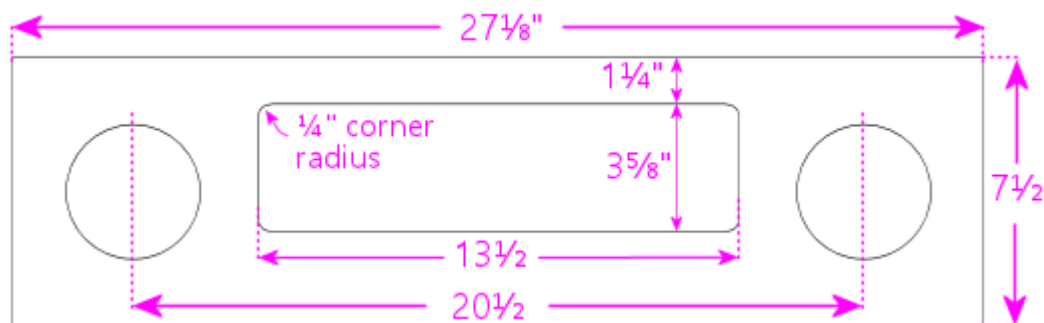
physically, but it also creates a cleaner appearance.

So if it's impossible to use the X-acto knife tracing technique, what can we do? There are a couple of potential ways to handle this, but really only one good way:

- Not so good: Print the whole decal on transparent film. This is what I thought of first when I was building my cab, but Brad Bowman (the printing pro who did my decals) set me straight on this, explaining why it's not a good idea. To do it right, you'd need the print shop to print an opaque white layer under your graphics, which most retail print shops can't do. Without an opaque white backing, the graphics would be invisible against a black MDF background, and washed out against a white MDF background. In addition, you don't want the sticker to cover the DMD opening anyway, even if it were transparent, since it wouldn't be as crystal clear as the acrylic.
- Good: Print on normal opaque stock, and have your print shop laser-cut the DMD opening out of the decal during production. This is the right solution, and most print shops can do it. You give them a cutting template and they'll make the cutouts as specified. You can have them laser-cut the speaker openings at the same time so that you don't have to cut those out by hand later. I used this approach for my speaker panel decal and it worked perfectly.

Applying a decal with cutouts is easier than you'd think. The trick is that the print shop will install a "mask" on the visible side of the decal. The mask is a layer of adhesive paper (similar to masking tape) that covers the whole decal, including the cutout areas. It makes installing a decal with cutouts just like installing a decal without any cutouts. Once the decal is in place, you peel off the mask, and what's left behind is the decal with your cutout areas removed.

Here's the cutting plan for the decal. This is based on the plans we provide later in this section, so it'll work if you're fabricating your own panel based on our plans. If you're using pre-cut panels from VirtuaPin or another vendor, I'd take measurements from your physical panel instead of relying on this, since there might be some variations in different vendors' versions of the panel.



Some notes:

- Center the speaker cutout circles vertically
- The speaker cutout circles should be exactly the same size as the ones used in your MDF panel and acrylic facing (the usual sizes are $3\frac{3}{4}"$ diameter cutouts for 4" speakers, and $4\frac{3}{4}"$ diameter cutouts for 5.25" speakers)
- Center the speaker cutout pair left-to-right in the overall decal width
- Center the DMD cutout left-to-right in the overall decal width
- Many printers will want you to include some extra padding around the perimeter of your design, such as an extra 1" all the way around. Adjust the cut positions accordingly.

Painting

Before assembling the panel, you should paint the MDF panel in the desired finish color. The most important areas to paint are the inside edges of the DMD and speaker openings, since those are visible through the cutouts. Everything else is hidden - the whole front is hidden by the plastic facing, and the back is inside the machine. I found it easiest to paint the whole front side, getting the edges of the cutouts while I was at it. This also ensures that you won't see any raw MDF poking out around any of the openings. The original Williams panels are painted flat black, which helps avoid any reflections around the display.

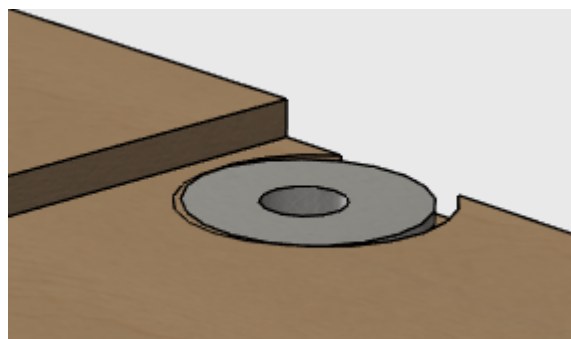
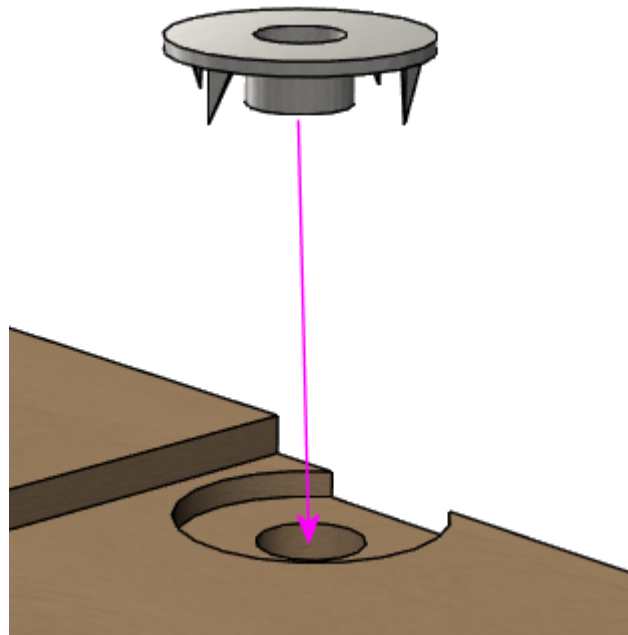
I think it's better not to paint the back side, in case you want to use any adhesive fasteners for anything attached to the back.

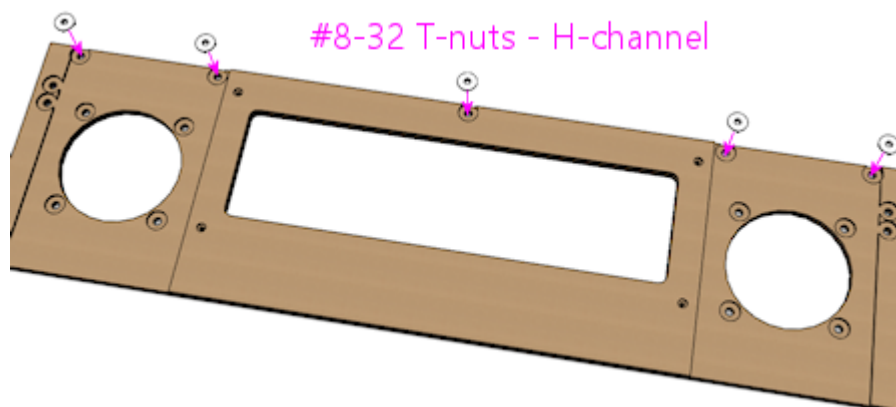
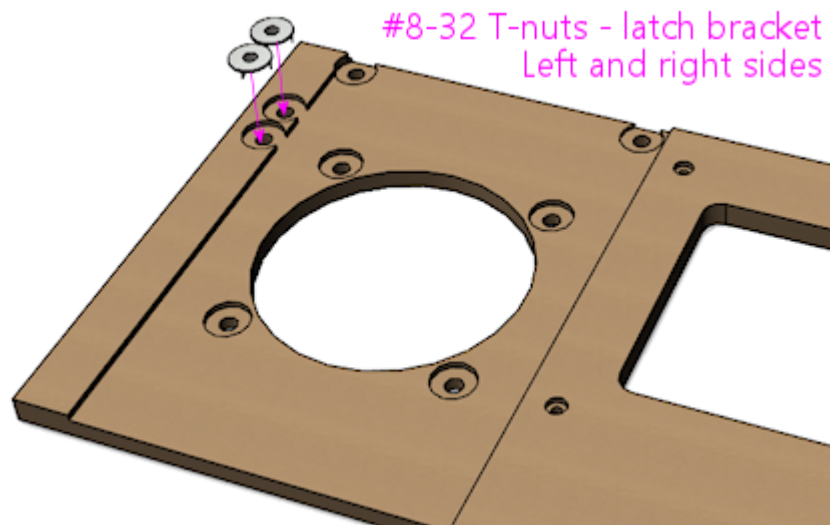
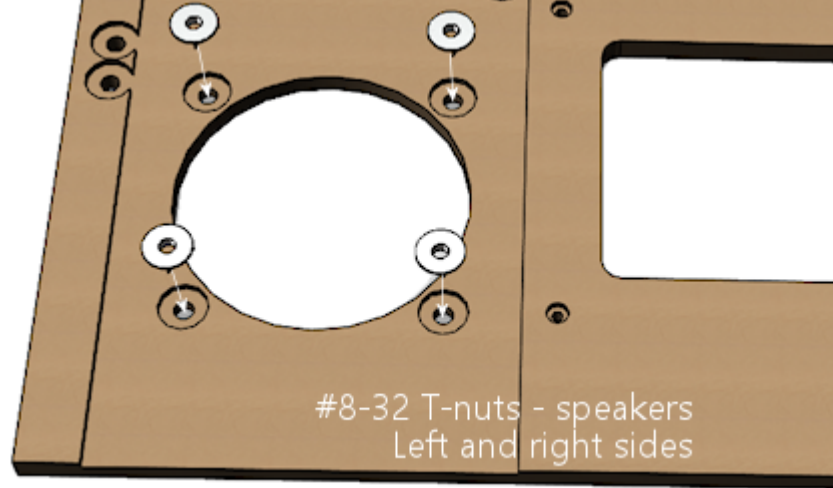
Assembling the panel

Here's the procedure for assembling the panel from parts. If you bought a partially assembled panel from VirtuaPin, just skip the parts where we talk about things they've already installed for you.

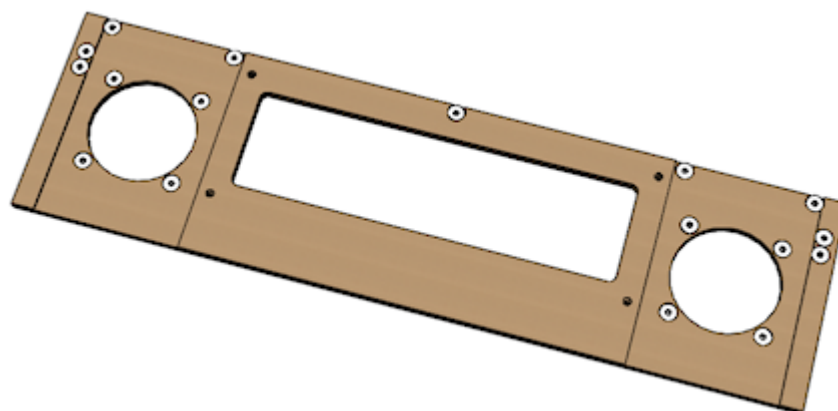
Install the T-nuts

Install #8-32 T-nuts for the speakers, latch brackets, and H-channel, as illustrated below. The T-nuts are all installed from the **front** side of the panel. The front is the side with the insets around the speakers.





When all of the T-nuts are installed, it should look something like this:



Install the DMD screws

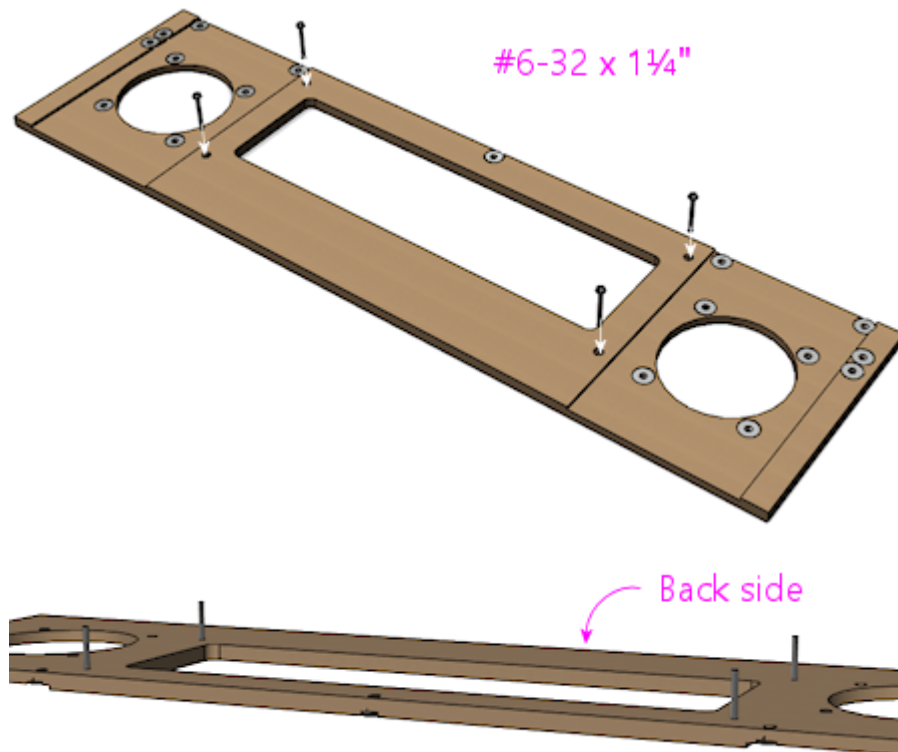
The original WPC speaker panels used an unusual type of screw for the DMD attachment points, known as a "spiral fin shank" screw. These have flat nail-like heads, machine screw threading down most of their length, and wood screw threading at the top (head) end. This weird combination is designed so that you can permanently fasten the screw to a wood panel with the threaded end sticking out, to make an attachment point.



These are so minutely specialized that you can probably only find them from pinball vendors like Marco Specialties or Pinball Life. Look for Williams/Bally part numbers 237-5957-00 (#6-32 x 1-3/16") or 4506-01104-20 (#6-32 x 1-1/4").

If you prefer to avoid the special pinball parts, you can make this work with ordinary #6-32 x 1-1/4" machine screws. It takes a little extra improvisation, which we'll explain below.

With either type of hardware, you insert screws from the **front** side of the panel, as illustrated below.



Note how the screws go in through the front side of the panel so that the threaded ends stick out the back. These screws are going to serve as mounting posts for the DMD device or laptop display screen. The DMD (or mounting bracket for your video panel) has holes that will fit over these screws, and you'll use nuts to fasten it.

The arrangement with these screws is unusual in that you want them locked in place

on the panel. You'll use them to secure the DMD device, by fitting nuts onto the bolts - but to make this work, the bolts must be fixed in place so that they can't turn, because you won't be able to access the screw heads after assembling the panel. If you're using the special fin shank screws mentioned above, they're specifically designed to work this way, but if you opted for ordinary machine screws instead, they'll need a little help.

For the special spiral fin shank screws:

- Push the screw into the hole in the panel, tapping it with a hammer if necessary, until it's pushed through up to the "fin" portion
- Slip a #6 washer over the threaded end sticking out the back
- Fit a #6-32 nut onto the threaded end
- Tighten the nut, holding/pressing the head to keep the screw from turning
- As you tighten the nut, the fin threading will be forced to tap into the wood like a wood screw
- Keep tightening the nut until the fin is fully embedded in the wood and the head is flush
- Remove the nut and washer; the screw should stay securely locked into the panel by the fin threading

For ordinary #6-32 machine screws, the best way I can think of to keep the screws locked to the panel is to glue the heads into place with epoxy or other strong glue. I don't recommend using nuts or other fasteners instead, in part because they might get in the way of the DMD or video panel, and in part because they could loosen over time. You won't be able to access the heads after assembly because of the way the front plastic facing will be attached with adhesive, so the screws really have to be permanently installed. Here's the procedure I used:

- Mix up some epoxy, or use some other strong glue of your choice
- Insert the four DMD screws from the **front** side of the panel, so that their heads are all the way into the little recesses
- For each screw, back it out just enough to expose the head, put a generous dab of glue all around the head under it, then push it back in all the way in
- Keep the panel turned front-side-down while the glue dries, so that the glue won't ooze down the screw threads
- Put wax paper (or something that the glue won't stick) to under the panel while it dries, in case any glue drips

The idea is to glue all of the screws in place at the heads without spreading glue over the rest of their length. You still want to be able to fit nuts onto them to install the DMD panel, so you don't want any glue smearing onto the threads while you insert the screws or dripping down the threads after they're installed.

Install the speaker grills

Parts:

- For 4" speaker openings: Stern 535-8081-00, 535-8081-01. Available in a variety of colors, including metallic finishes.
- For 5.25" speaker openings: I can't find any speaker screens specifically for 5.25" speakers in the old panels. The closest thing is the speaker grills for the WPC-95 panels, Williams/Bally 04-10382-7-4. Those are for the 5.25" squarish openings in the newer panels, but they should also fit the round openings in

the old panels, possibly with a little trimming required. Another option is to get a pair of the 7" screens designed for use with the subwoofer ports in the bottom of the main cabinet (Williams 03-8603-1, 03-8603-3, 01-6733). Those are the same material, and it's easy to cut them to a smaller size with scissors.

- If you want to look for the raw materials, the screens are made from perforated plastic, black, 1/16" thick, with 1/8" holes on staggered 3/16" centers, 40% open. That turns out to be a common perforation pattern for both plastic and metal sheets (such as stainless steel or brass, which can look nice in some color schemes). You can find materials like this from bulk industrial supply vendors online. The snag is that those guys all sell in large sheets, which can be pricey. Expect to pay upwards of \$50 for a 24x24 inch or larger sheet of plastic, and more for metal sheets. The other snag is that, for whatever reason, it's hard to find the bulk plastic sheets in black - they're usually white or gray. But that might be just the thing for your color scheme.

The speaker grills install on the front side of the panel. Fit them into the recesses around the speakers.



The 4" type should come with holes pre-cut at the corners to match the screw holes, so align those with the T-nuts. For the 7" type, you'll have to cut holes at the corners to match the screw holes.

You might want to tape these in place with a little masking tape, to keep them from sliding around during the rest of the assembly procedure. The screws that attach the speakers will help hold the grills in place once they're installed, but we need something to keep them stationary until then. Make sure that the tape isn't be visible through the cutouts.

Install the acrylic facing

If you're planning to use decals with graphics, apply the decals to the acrylic before installing it.

The acrylic attaches to the MDF with double-sticky tape. (It seems jury-rigged, but that's the way they built many of the originals. Some apparently used glue instead, but I find the tape easier to work with.) Use a good quality foam tape, but not one of the super-strong permanent adhesives, because you might want to be able to pry this thing off at some point. There all of those buried parts that you can't get to any other way. And don't get carried away with vast amounts of tape. A couple of pieces of tape at the bottom corners should be enough. The whole top edge will be held in place by the "H" channel molding, so you don't really need any tape there.

Once the tape is in place, carefully align the facing with the MDF and press it onto the tape.

Install the latch brackets

The latch brackets are Williams/Bally part number 01-8535:





They install on the back of the panel at the top corners:

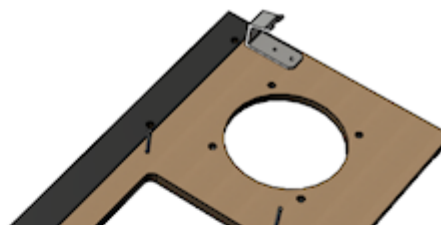
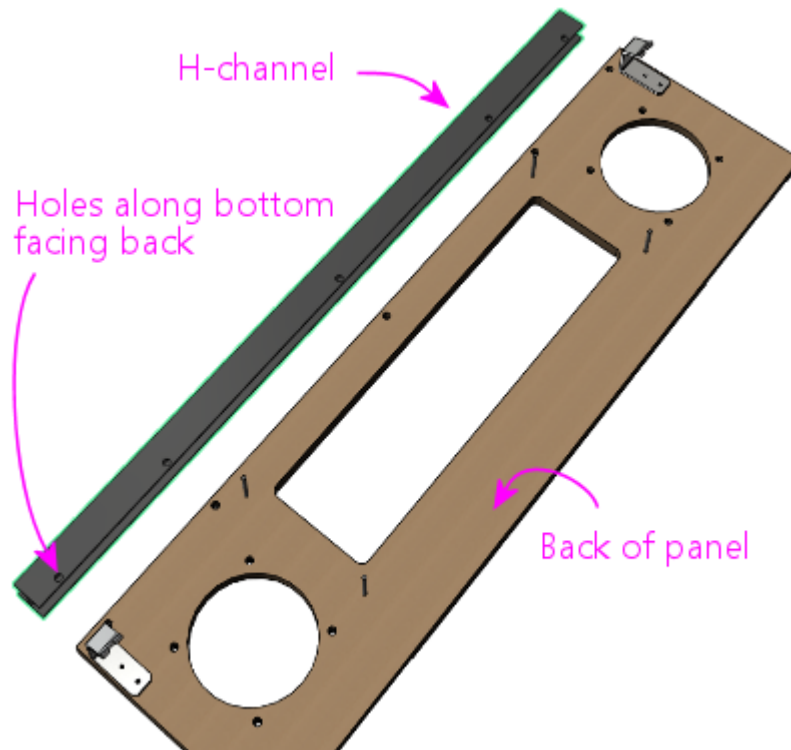


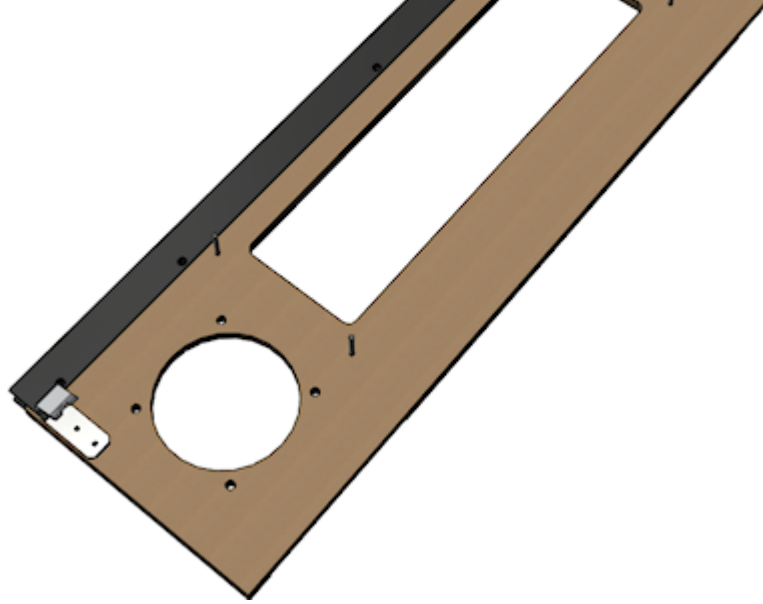
Install each bracket with two #8-32 x 3/8" countersunk machine screws:



Install the H-channel trim

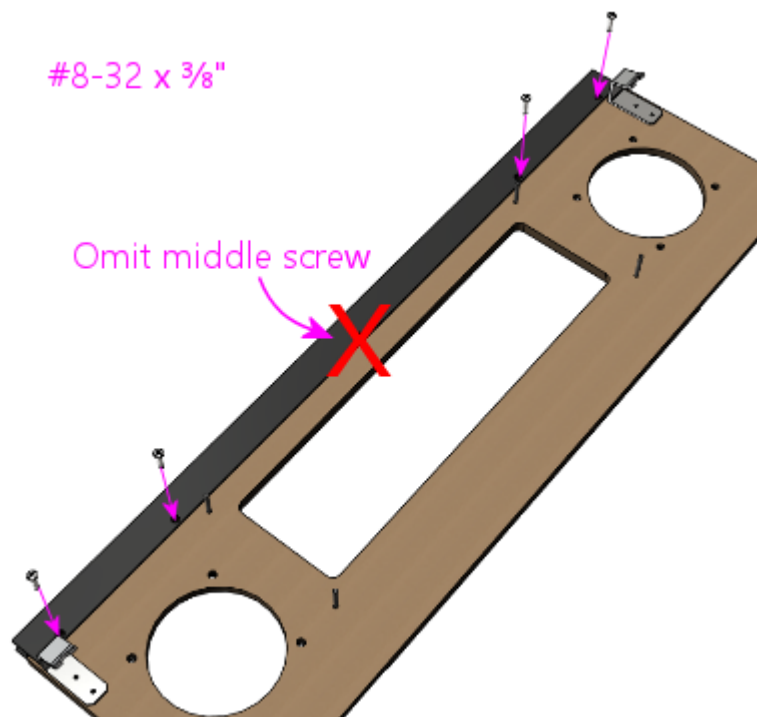
The H channel trim goes across the top of the panel. Orient the channel so that the screw holes are on the back side, on the bottom half, so that they align with the screw holes in the MDF.





Fasten the panel with four #8-32 x 3/8" pan-head machine screws. On the real machines, they also use external-tooth lock washers with these.

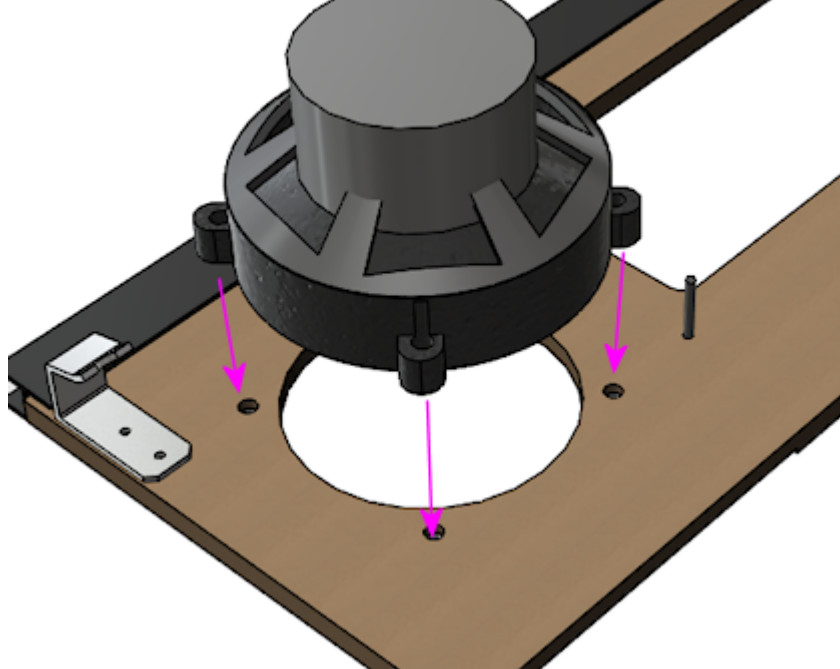
Note that the H channel and MDF panel both have holes for five screws. You should **omit the middle screw**. The Williams panels also usually left out the middle screw, because it's too close to the DMD. This is even more important with an LCD video display, since it'll extend a little past the top of the H channel - if the screw is there, it can come into contact with the face of display and scratch it. Better to leave it out to avoid that.



Installing the speakers

This part should be easy. Your speakers should have mounting holes arranged in a square pattern around the perimeter of its base plate. Assuming your speakers are the right size for your panel, these should line up with the four pre-installed T-nuts around the perimeter of each speaker opening in the panel.





Fasten with #8-32 screws. You might need to try different lengths to find the right length, since it depends on how tall the base ring around the speaker housing is. Make sure the screws aren't so long they hit the acrylic on the other side; you can always add some washers to take up extra length if needed.

The real machines use external-tooth lock washers with the screws to help secure them. This is to help prevent the screws from working themselves loose from vibration - and obviously, a speaker's whole job is to produce a lot of vibration.

If your speakers don't fit the panel, you can either replace them with speakers that do fit, or you can create an ad hoc adapter from another piece of MDF or plywood. To create an adapter, cut a piece of MDF slightly larger than your speaker's footprint, with a circular cutout large enough for the speaker aperture, and mount your speaker to that. Drill holes in the adapter to match the speaker mounting holes in the speaker/DMD panel, then mount the adapter to the panel.

Installing speaker LED strips

One of the popular mods with the pinball collectors these days is speaker lighting, usually via LED strips placed around the perimeter of the speaker openings.



You can buy retail kits to retrofit this kind of lighting into the real machines. One of

those would easily adapt to a virtual cab since we're using identical speaker panels. But as long as you're assembling your own panel, it's actually pretty easy (and inexpensive) to add lights yourself.

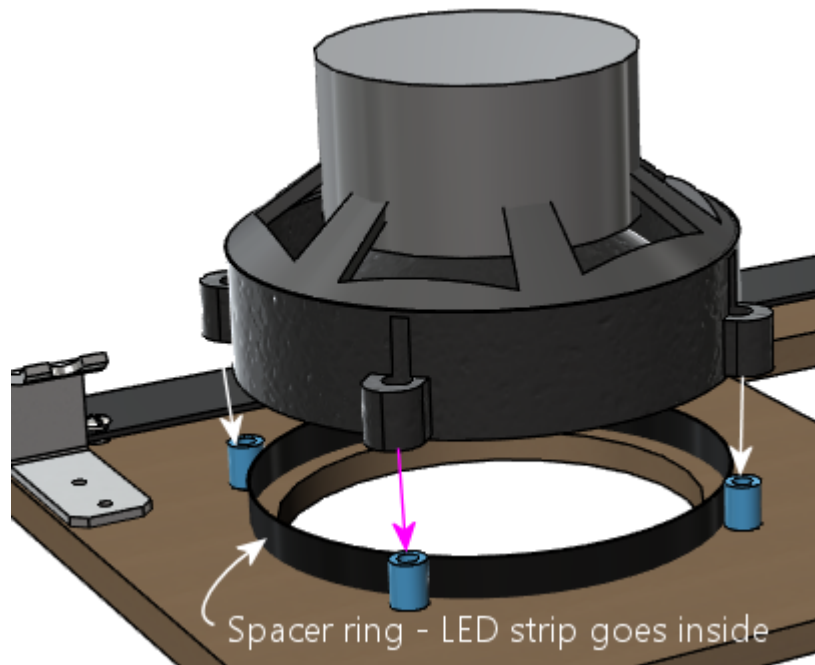
The parts required are:

- The common "5050" RGB strips sold on eBay, RGB, 1cm wide, 60 LEDs per meter (the exact same type used for Chapter 58, undercab lighting). You need a length equal to the twice the circumference of your speaker openings; about 1 meter should be plenty.
- One "Mini Controller for RGB LED light strips" from eBay; many types are available, including remote controlled and Wi-Fi, but the cheap "mini 3 key" type (about \$3-5) like the one pictured at right is more than adequate for this job
- 1cm nylon spacers for your speakers, quantity 8
- Stiff cardboard or similar material, black



If you're lucky, you won't even need the spacers and cardboard. The 4" speakers I got from Flipper Fidelity have a built-in 5/8"-deep spacer ring at the front, which is perfect for this - I simply stuck my LED strips around the inside perimeter of the spacer ring.

But most speakers don't have any sort of spacer at the front, so you have to fashion your own. Cut a 1 cm-wide strip of the cardboard or other material you're using as the spacer - equal to the width of the LED strip. Form this into a circle. Stick the LED strip to the inside. Position it over the speaker opening in the panel. Use the nylon spacers to install the speaker just behind the ring.



Alternatively, you might be able to stick the LED strip directly to the perimeter of the speaker cutout circle in the MDF itself. The only snag is that the MDF isn't as thick as the LED strips are wide, so it's not a perfect fit. The LED strips are 1 cm wide, and the MDF is only about 7 mm thick here because of the speaker grill routing on the front, so the LED strip will stick out beyond the back of the panel by about 3 mm. If your speaker's cone is a little recessed, the excess might happen to fit into the recess, in which case you're done. But if there's not room, you can add 5 mm nylon spacers between the speaker and the panel, or you can just use a few washers to

serve as spacers.

Wiring the speaker LEDs

The red and black wires coming out of the mini-controller have to be wired to 12VDC power, so you'll need to run some hookup wire into your cab where you can get to a 12V power supply. The LED connection to the mini-controllers usually use those little 4-pin LED strip connectors, which you can also find on eBay. See Chapter 58, Undercab Lighting for information on the peculiar little connectors used with these light strips - since these are the same type of light strip, you should be able to cobble together the right set of connectors by consulting that section.

It can be a little difficult to manage the wires attaching to the LED strip. I don't have any particular tricks to suggest; just be patient until you find an arrangement that works. You should secure the wiring to the back of the panel with a couple of wire clips once you have a good setup so that you don't pull it loose in future work.

I simply wired the strips in my two speakers together with hookup wire, so that they both run off of the same controller. The controller I have remembers its last mode setting, so it's a set-and-forget sort of thing. You just click through the modes until you find one you like, and leave it there; that same mode will then be used each time you power up the machine. The mode I use on my machine is a slow color rotation.

If you prefer, you can skip the mini-controller entirely, and instead use your feedback controller and DOF to control the speaker lights. This is just a matter of wiring the speaker LED strips exactly like the same way you wire your undercab strips, as explained in Chapter 58, Undercab Lighting. The DOF config tool doesn't have a pre-programmed toy for "Speaker Lights", so the simplest thing to do is wire the speaker lights directly to your undercab lights, so that they use the same DOF color effects.

Foam tape around the DMD opening

This is optional, but on the real machines, they installed a ring of 1/4"-thick black foam tape around the DMD opening on the back of the panel. Common weatherstripping tape works perfectly, and it's cheap. To match the original equipment, use tape that's 1/4" thick and 3/4" wide, self-adhesive on one side. (Don't use double-sticky tape - you don't want it to stick to the display device.) Simply affix a strip on each edge of the DMD opening, flush with the edge. Black tape is best because it'll be essentially invisible and won't reflect any glare from the display, and it's the easiest thing to find anyway.





Use black foam weatherstripping tape around the edges of the DMD opening, 1/4" thick by 3/4" wide. This protects the display panel from scratches and creates a little gap between the display and the back of the MDF, which is especially helpful for LCD panels.

They used foam tape on the real machines with the original amber plasma DMD devices, but it happens to be even more beneficial with LCD panels. The tape protects the display from scratches, reduces rattles (there are speakers attached right next to this thing, after all!), and it spaces things out a little bit. The extra space is especially helpful with LCDs because of the way they stick up slightly above the top of the MDF panel. The extra space keeps the panel from pressing against the top "H" channel trim.

Installing a real DMD

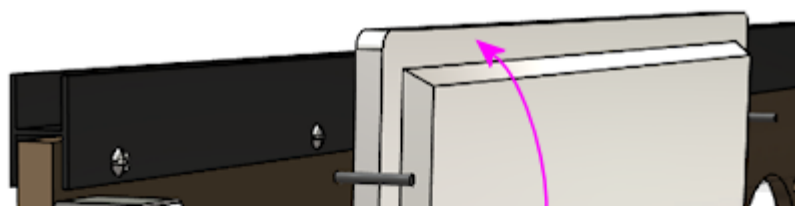
It's easy to install a real DMD device in the standard panel, since the layout of the "posts" you installed earlier is designed specifically to fit the mounting holes on the DMD. All real DMD devices should use exactly the same mounting hole pattern that they used on the original displays from the 1990s.

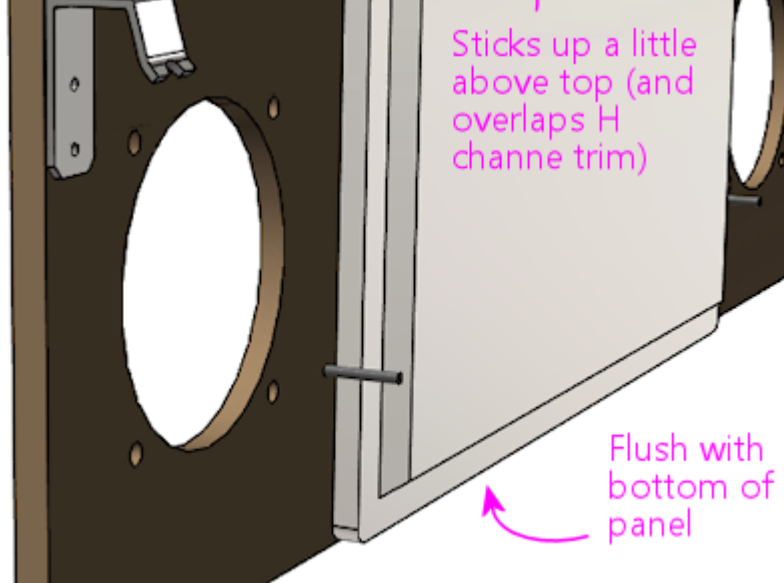
Your DMD device should have corner mounting holes that line up with the four "posts" sticking up out of the back of the panel - those 1 1/4" screws that we epoxied into the panel earlier. Fit the DMD mounting holes over the posts and seat the front of the DMD flush against the panel. Use #6-32 nuts with lock washers to fasten it.

Installing a video panel for the DMD

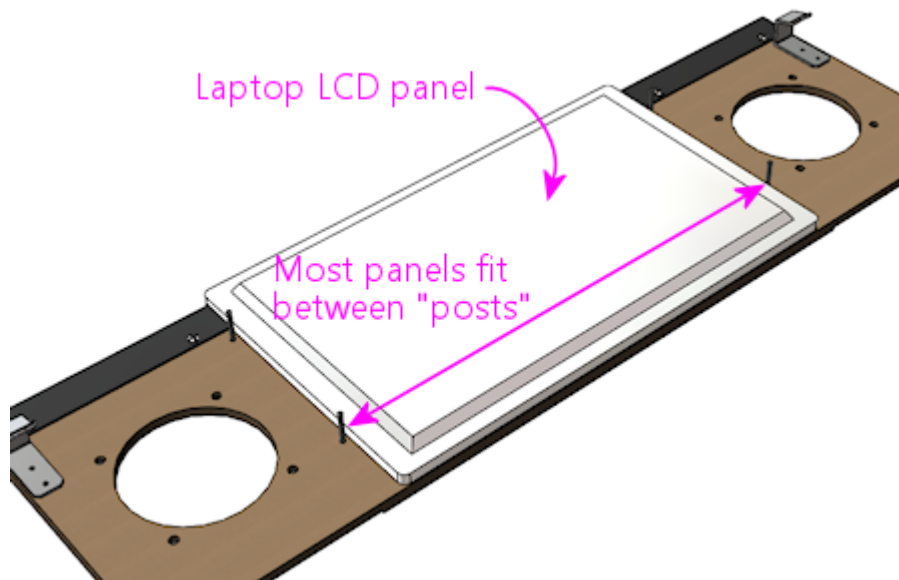
Installing a laptop LCD panel in place of a DMD device takes some improvisation, since the speaker panel wasn't designed with this in mind. There are some slight mismatches that we have to work around.

The first mismatch is that the laptop panel will probably be too tall. A 15.6" 16:9 LCD display is a perfect fit for the width of the DMD opening, but it's much taller than the opening, and it's even a little taller than the speaker panel overall. Being taller than the opening is no problem; when you set up the window layouts in your various pinball programs, you can just move the DMD window so that it's in the visible portion of the display, and leave the hidden portion of the display unused. Being taller than the whole panel is a little more of a problem in that a small portion (usually less than an inch) will stick up above the top of the panel. But that's usually not a problem. If you're using a translite, the translite will sit directly above the top of the panel, resting in the H-channel, so the excess portion of the laptop display will be hidden behind the translite. There's usually nothing else occupying this space, so there shouldn't be any conflicts. So the height mismatch, at least, is one of those things that looks like it might be a problem, but ends up solving itself.





The second mismatch, which does require some work on your part, is that the LCD panel won't have any mounting hardware that lines up with the "posts" on the speaker panel. A suitably sized panel should just fit between the posts side-to-side.



So you have to come up with your own mounting apparatus. It's not hard, but I don't have any "standard" solution to suggest. I used sheet metal to fashion some brackets that fit over the "posts" and hold down the edges of the panel, securing these with nuts on the posts. If you want something more off-the-shelf, you might try looking at picture/mirror hangers at a hardware store - something like a mirror clip would probably work.

Speaker panel construction plans

The original Williams speaker/DMD panel was designed to hold an asymmetrical speaker pair, with a large 5.25" speaker on the left and a small 3" speaker on the right. The speaker cutouts and mounting holes in the panels were sized to match. The different speakers were used because, contrary to appearances, they were *not* a left/right stereo pair. They were in fact a midrange/tweeter pair, and were wired for monophonic playback.

In contrast, most virtual cab builders today use a pair of full-range car speakers, and

connect them as a stereo pair. As a result, we want the two speaker openings to be the same size, unlike the original Williams panels with their asymmetrical cutouts.

Fortunately, it's easy to adapt the basic speaker panel plan to whatever mix of speaker sizes you prefer. Everything about the plan stays the same except for the size of the speaker cutouts and the positions of the screw holes for mounting the speakers - those obviously have to be changed to match the speakers you're using.

So we'll start with the base plan, with all of the specs that are independent of speaker size. Then we'll provide two sub-plans for the most commonly used speaker sizes: 4" and 5.25". Those are the sizes that almost everyone uses because they're standard sizes for car speakers, which means that lots of options are available.

Materials

- The main panel is **3/8" MDF** (medium-density fiberboard)
- The plastic cover on the original equipment was 2mm PETG, but most virtual cab builders use **2mm clear acrylic**, since it's easier to find
- Optionally, a decal printed with custom artwork

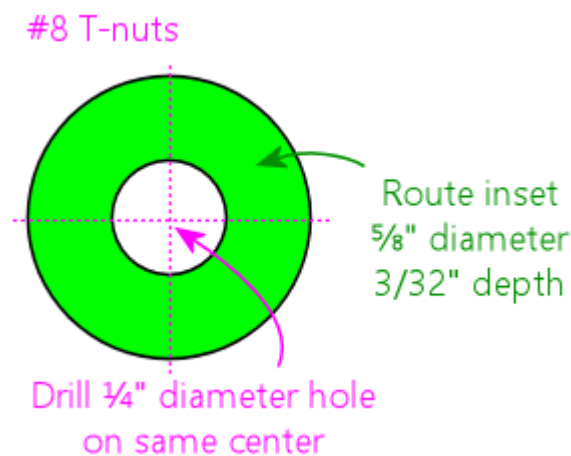
The original equipment had the artwork screen-printed directly onto the PETG cover, but that's not practical for most DIYers. You can easily have a custom decal printed, and it'll look just as good.

Base plan

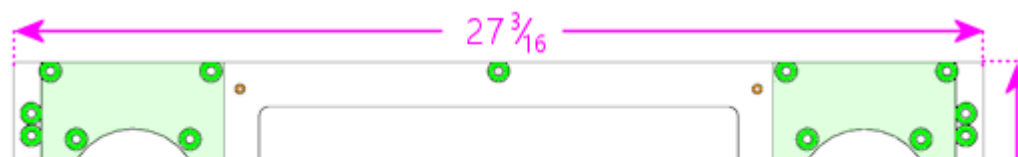
Everything in the base plan is independent of speaker size, so start with this regardless of what type of speakers you're using.

All of the routed insets (shown in the diagrams below as green and orange areas) are on the same side, which will be the front of the panel when it's done.

The panel is symmetrical left to right (the left side is a mirror image of the right side), so everything in the panel should be referenced to the center point. This will be especially helpful if you're using a custom width.

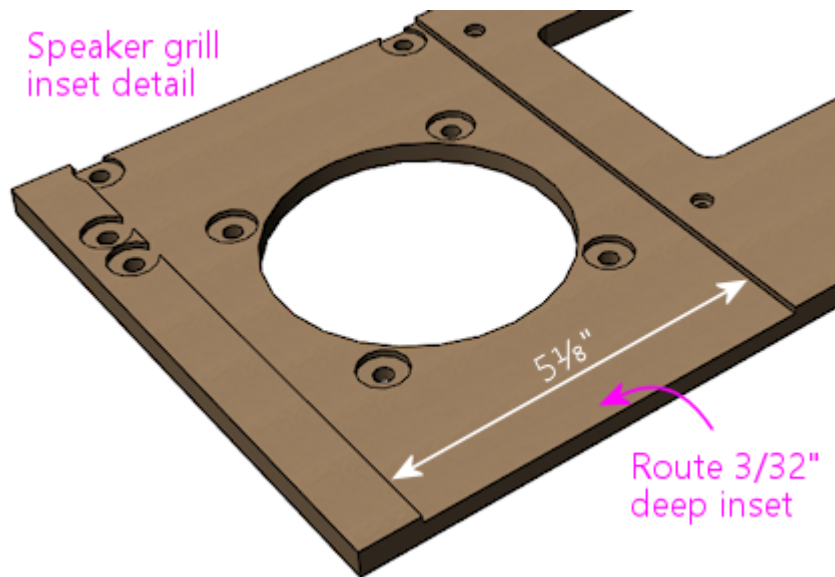
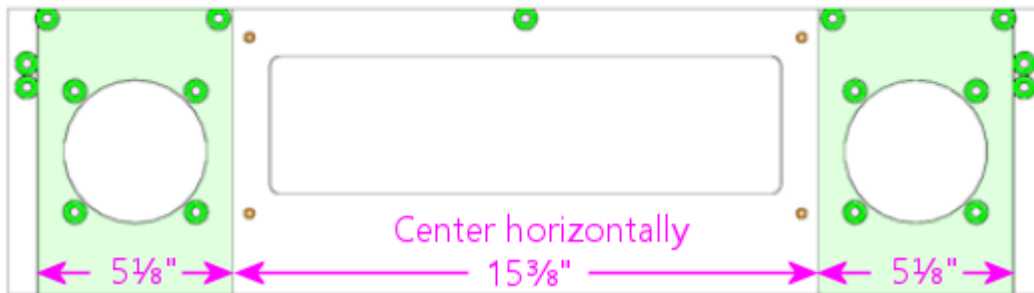


All illustrations below show the front side of the panel (the side with the acrylic overlay, facing the player).

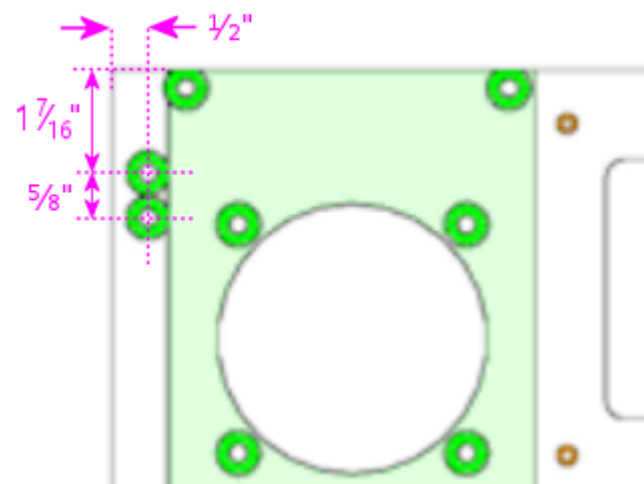




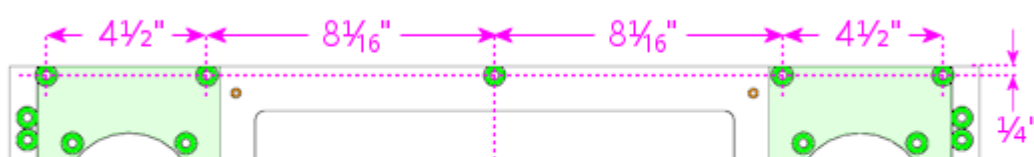
Insets for speaker grills
Route 3/32" deep insets light green areas

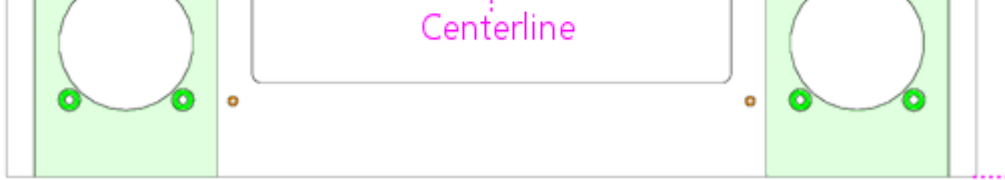


Latch bracket T-nuts - #8-32

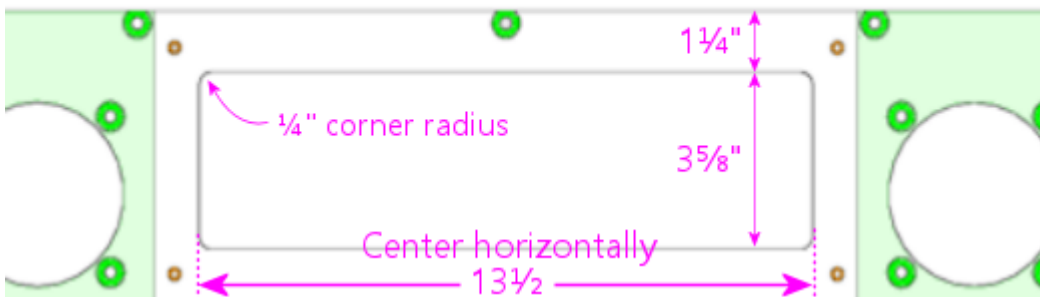


H-channel T-nuts - #8-32

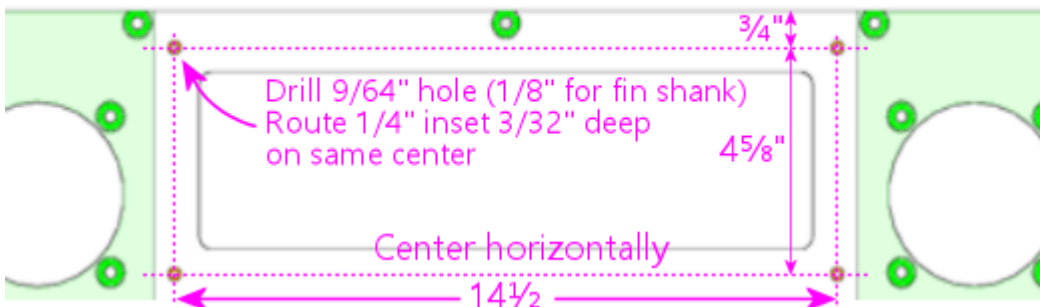




DMD Cutout



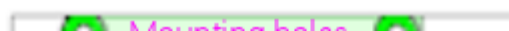
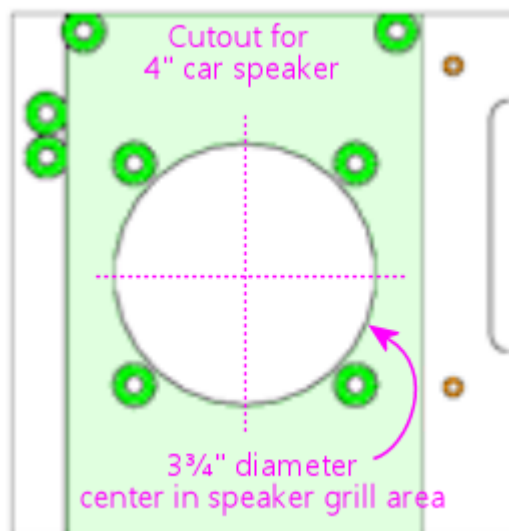
DMD mounting holes - at corners of DMD opening

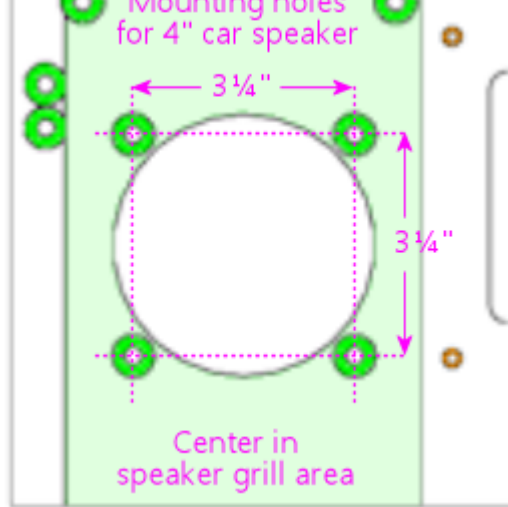


Note: if you're using the special spiral fin shank screws for the DMD attachment posts, drill $\frac{1}{8}$ " holes; if you're using ordinary machine screws, drill $\frac{9}{64}$ ". The smaller hole will help the fin shank screw fasten more securely.

Cutouts for 4" speakers

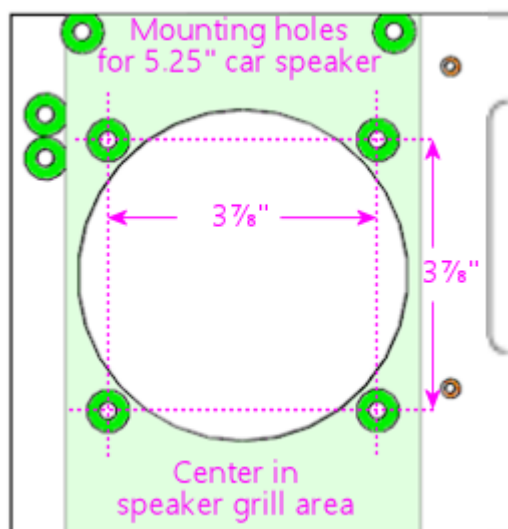
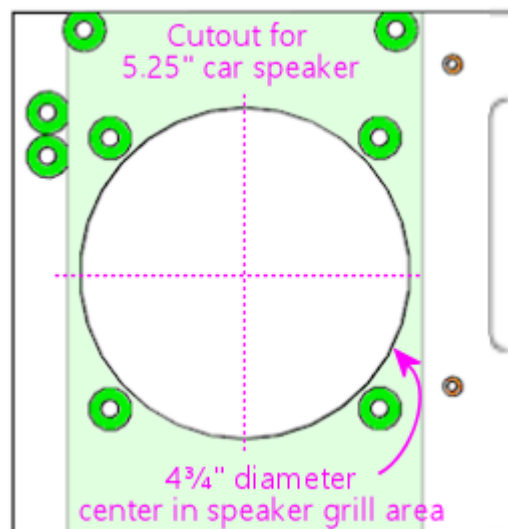
Use these speaker cutouts if you're using speakers that fit the standard 4" car speaker design. Everything else is the same as in the base plan above. Center the cutouts in the speaker grill area.





Cutouts for 5.25" speakers

Use these speaker cutouts if you're using speakers that fit the standard 5.25" car speaker design. Everything else is the same as in the base plan above. Center the cutouts in the speaker grill area.



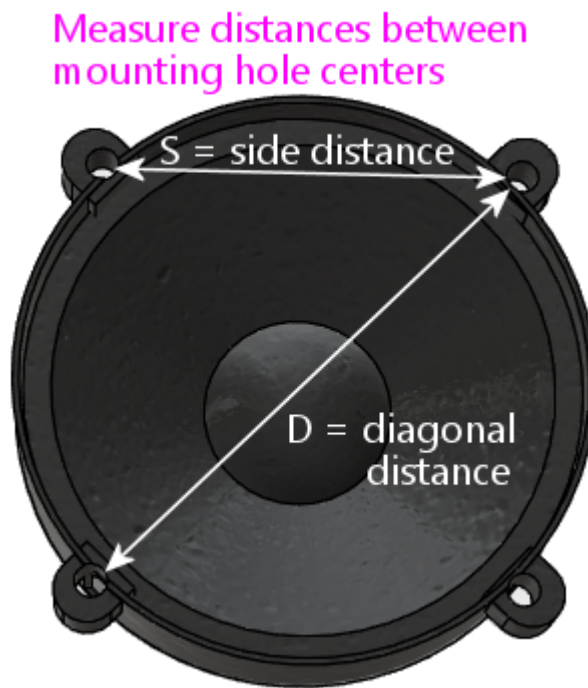
Cutouts for custom speaker sizes

Since the base plan is independent of speaker size, it's easy to use any non-standard

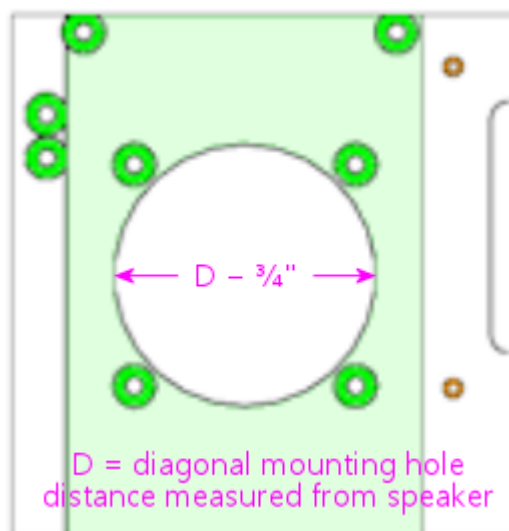
speaker size.

To create a custom plan:

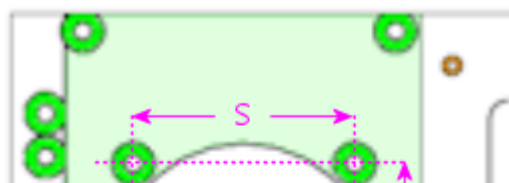
- Start with the base plan
- Measure the distances between the **centers** of the mounting holes on your speaker, as shown below:

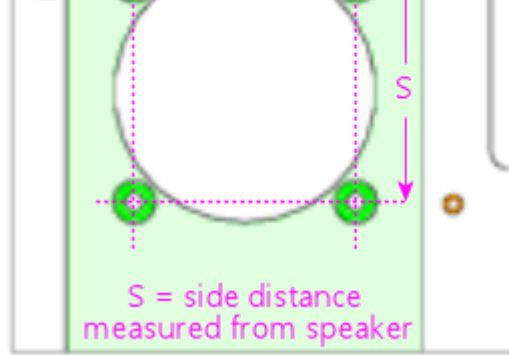


- For the circular cutout, use a diameter of **D minus 3/4"**
- Center the circular cutout in the speaker grill area



- Use the speaker as a template to mark the positions of the mounting holes, centering it on the circular cutout, or use the measured side distance "S":





- Drill and route the mounting holes using the normal #8 T-nut scheme from the base plan

Front facing

The plastic facing, thankfully, has a much simpler cutting plan than the MDF. The only cut lines needed are the exterior outline and the two speaker opening cutouts.



Center the speaker cutouts vertically, and center the pair horizontally, with the spacing between their centers as shown.

The cutout circles should be exactly the same size as the speaker cutout circles in the MDF panel. The usual sizes are $3\frac{3}{4}$ " diameter cutouts for 4" speakers, and $4\frac{3}{4}$ " diameter cutouts for 5.25" speakers, but go with whatever you used for the MDF panel.

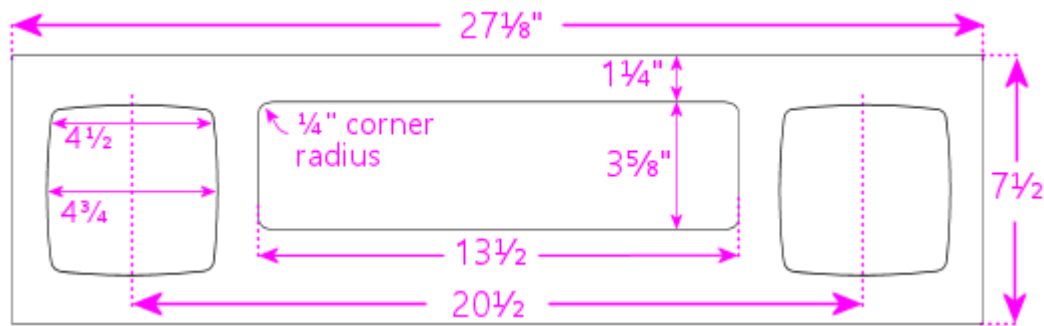
The width we're using for the plastic facing is a hair less than the width of the MDF panel ($1/16$ " less, to be precise). This is intentional, and it also seems to be what Williams did, at least for the panels I've measured. My rationale for making it slightly narrower is to allow a little room for error in case you don't get the width cut perfect on both pieces. The MDF is sized to be a tight fit to the backbox, so if the facing were to overhang the MDF on either side, it might not fit at all. Cutting the facing a hair narrower than the MDF makes it a little more forgiving, especially considering that you also have to line up the speaker cutouts on both pieces.

WPC-95 style facing

If you want to emulate the Chapter 33, WPC-95 style using the original style's MDF construction, you could substitute matte black acrylic for the clear acrylic, and cut out the DMD opening in the facing. (With clear acrylic facing, the DMD is normally left covered by the acrylic, and that portion is left transparent rather than being covered by graphics.)

One visual difference with the WPC-95 style facing is that the speaker cutouts in the front facing are square rather than round. Approximately square, anyway; the sides are a little rounded, as illustrated below.

Here's a cutting plan for the acrylic layer using this design.



Use that for the acrylic facing only. For the MDF portion, use the base plan with 5.25" circular speaker cutouts.

Note that this still won't look exactly like a true WPC-95 speaker panel. The plastic on those panels is textured to give it a satin finish, which you wouldn't be able to reproduce precisely with acrylic. In addition, the edges of the speaker and DMD cutouts are beveled with a rounded fillet; the edges will be squared with laser-cut acrylic. But those differences are very minor; only a real pinball nerd would ever notice them.

Customizing the width

If you're using a custom backbox size, adjust the overall width of your panel to just slightly less than the inside width of your backbox - 1/16" to 1/8" less, depending on how tight a fit you want. For comparison, the original Williams speaker panels are about 1/16" less than the inside width of the standard backbox.

Here's how to adjust the other elements when changing the width:

- Keep the DMD cutout centered, and at the same height
- Keep the latch bracket T-nuts at the same distances from the outside edges
- Keep the H-channel T-nuts at the same distance from the top edge
- Place the middle H-channel T-nut on the left/right centerline, and position all of the other H-channel T-nuts based on their distances from the middle one
- The standard H-channel trim will fit fine on a wider-than-standard panel; it'll just leave some uncovered space at the sides. You'll have to cut the H-channel for a narrower-than-standard panel. Cut equal amounts from either side so that the screw holes remain centered.
- Adjust the speaker positions left-to-right for the desired appearance; nothing else depends on these positions, so you can put them wherever you like as long as they don't conflict with the DMD, latch brackets, or H-channel

33. WPC-95 Speaker Panel

This chapter goes into the details of the WPC-95 style of speaker/DMD panel, which is the type used in Williams and Bally machines from about 1995 and later. This type of panel is formed from a single piece of injected-molded plastic, in matte black, usually with a Williams or Bally manufacturer logo in metallic paint.



These panels didn't have any artwork apart from the manufacturer logo, so they're interchangeable across games from the WPC-95 generation.

There's a different style used in Williams/Bally machines made before 1995, which had an MDF core and plastic facing with screen-printed graphics specific to each title. See Chapter 31, Speaker/DMD Panel for more about the two panel types, and see Chapter 32, Original WPC Speaker Panel for details on fabricating, assembling, and installing the older style.

Buying or building

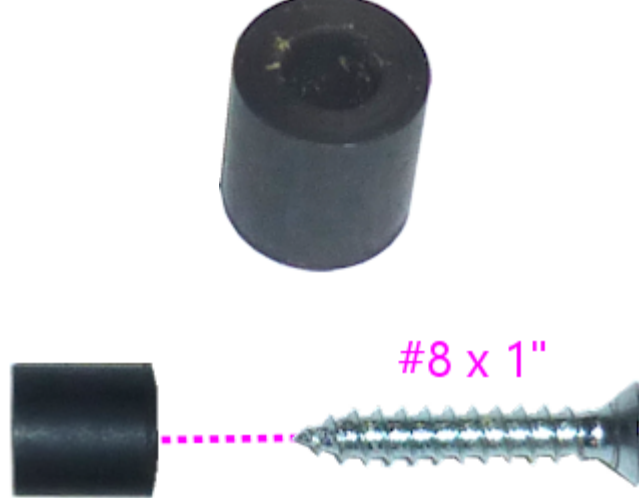
It's probably impractical to fabricate your own version of this type of panel. The originals are single-piece injection-molded plastic, which is a process that's very cheap if you're making a lot of copies of something, but expensive if you're just making one. 3D printing can often come to the rescue in a situation like this, but in this case it's probably not workable, since the part so large - 27" wide. A commercial 3D-print service might be able to fabricate an item this large, but it's likely to cost at least twice as much as buying the factory part.

If you really want to create your own panel of this style from scratch, I think you're better off not trying to duplicate the exact construction, but instead faking it a little bit, by building one that *looks like* this style but uses the construction techniques of the Chapter 32, older panel style. You can use the old design to make a panel that looks very much like the WPC-95 style simply by using a matte black plastic piece for the front facing. It'll still be a little different in the details, mostly in that it will lack the beveling around the speaker and display cutouts, but you'd have to look very closely to notice.

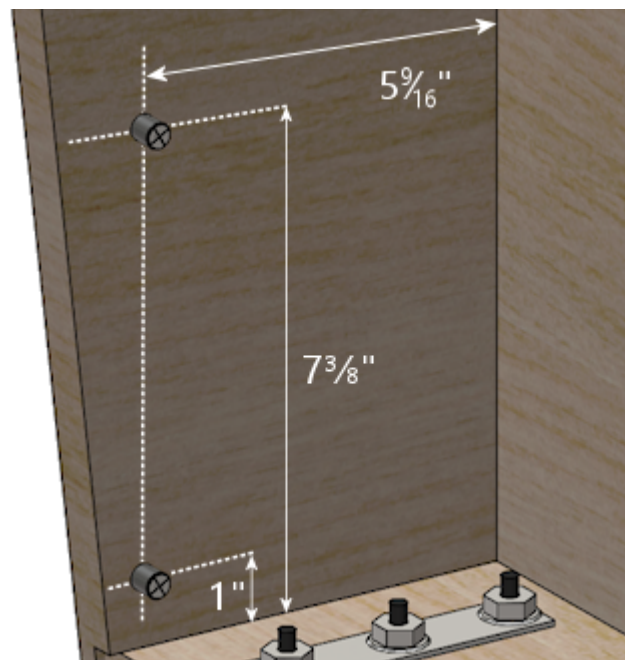
If you want an exact match to the WPC-95 style, you can simply buy one as a replacement part from a pinball supply vendor, such as Marco Specialties or Planetary Pinball. Replacement panels are available with Williams or Bally logos pre-printed in silver or gold. Look for part numbers 04-10382-7A, 04-10382-7B, 04-10374-7A, 04-10374-7G: those are all the same except for the logo selection. They run about \$100.

Backbox preparation

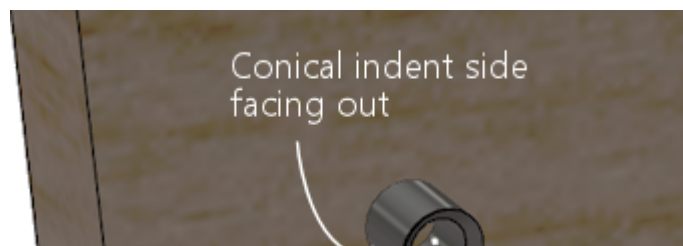
The WPC-95 panel installs in the backbox via "bushing buttons", Williams/Bally part 02-5223. You need four of these. These install with #8 x 1" flat-head wood screws.

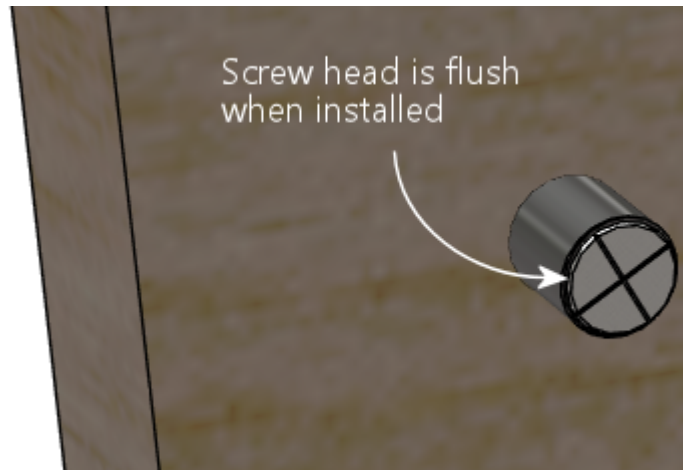


Install on the inside of each side wall (left and right) of the backbox, as shown below:



- Locations are to the centers
- Drill a small pilot hole for each screw
- The lower bushing is 1" above the floor, 5-9/16" from the back wall
- The upper bushing is 7³/₈" above the floor, 5-9/16" from the back wall
- Screw in each bushing with a #8 x 1" wood screw
- Orient the bushing so that the side with the conical indent for the screw head faces out - the screw head should recess into the indent so that it's flush with the bushing once installed
- Install the other two bushings on the opposite side

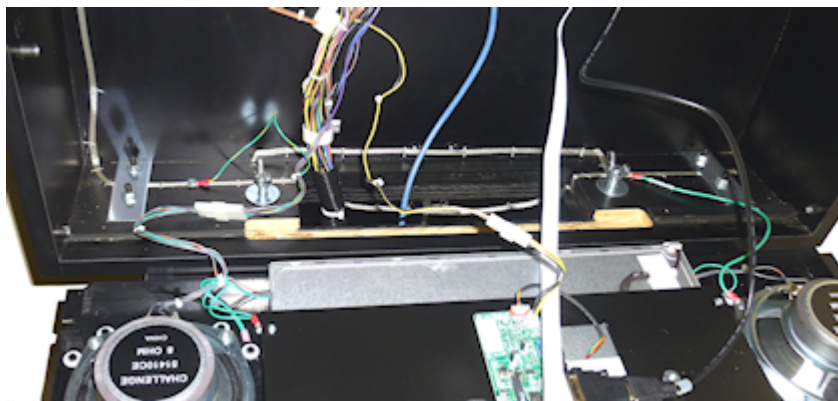




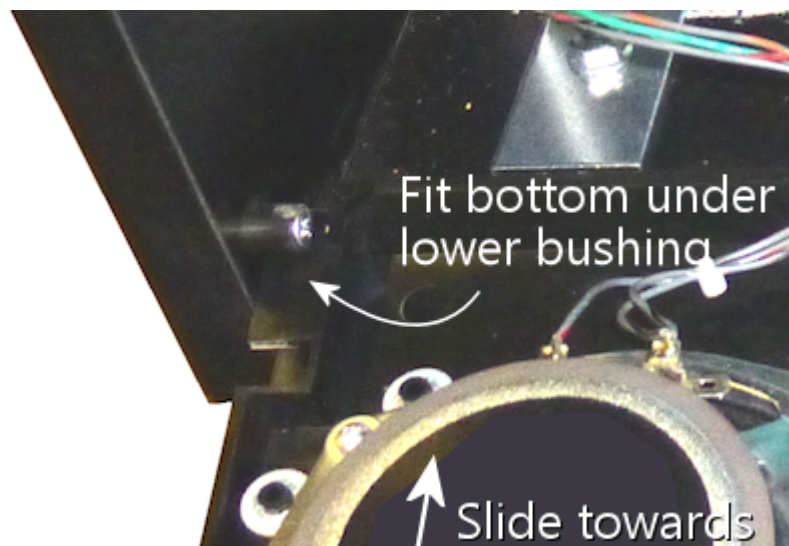
How to insert and remove the panel

The panel has slots on the back that fit over the bushing buttons.

To install the panel initially, lay it flat with the back facing up.

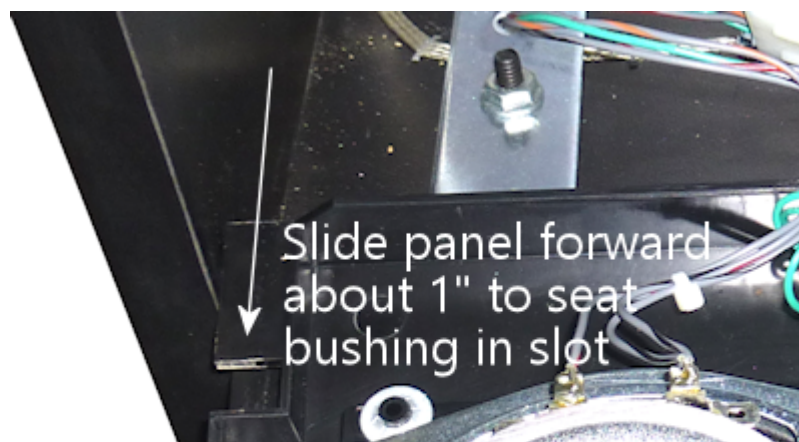
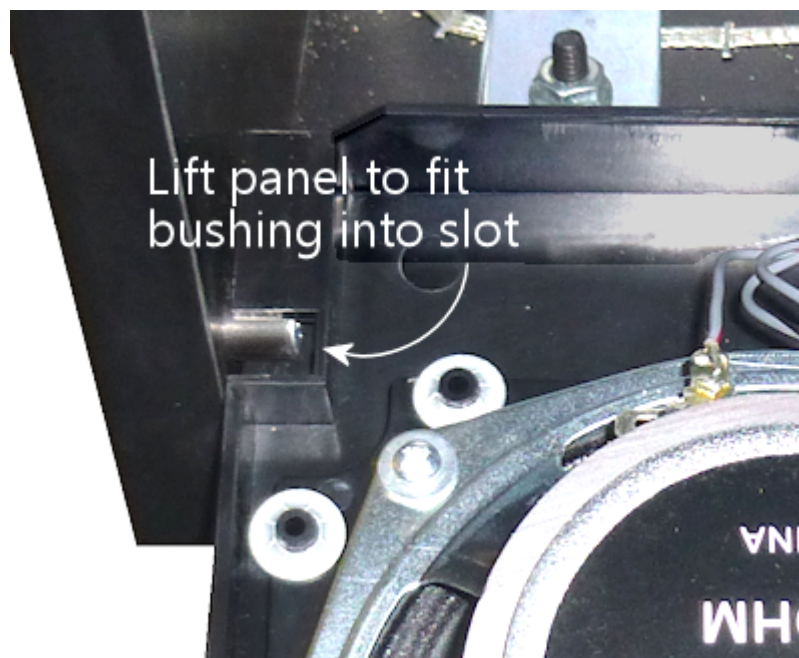
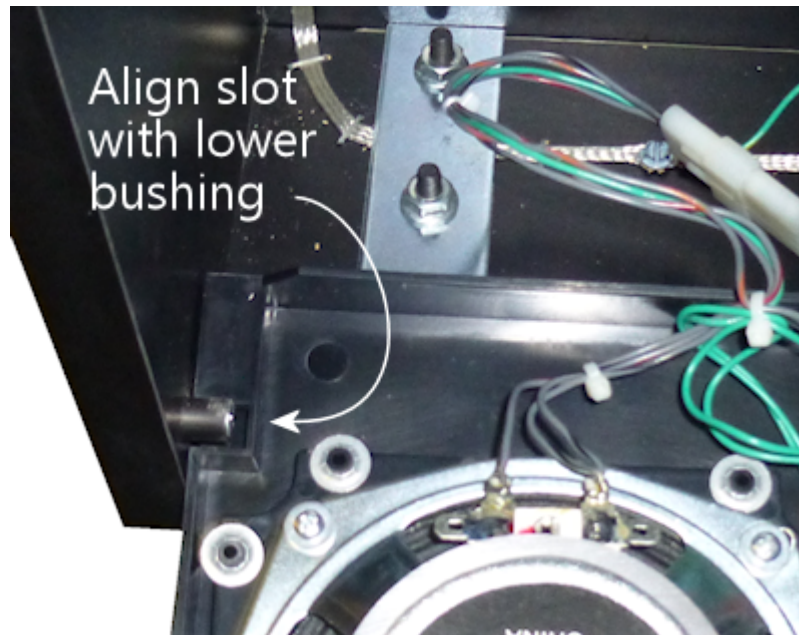


Fit the bottom edge *under* the lower bushings.





Align the bottom slots in the panel with the lower bushings, then lift the panel to fit the bushings into the slot. Pull the panel forward about an inch to seat the bushings in the slot.





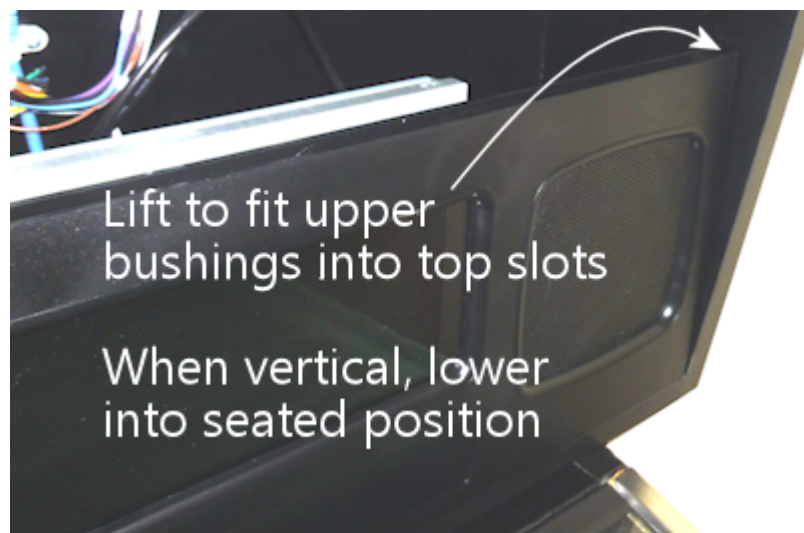
The panel is now installed! You can remove it again at any time by reversing the process to free it from the lower bushings and slide it out of the backbox.

The current position, with the panel attached to the lower bushings and folded down, is the normal service access position. You can fold it down like this any time you need to access the back of the panel, or any time you need to access the interior of the backbox behind the panel. This ability to fold the panel down without removing it is a convenience feature for the repair staff - one of the little improvements in the WPC-95 design.

To finalize the installation and get the panel into its proper position, simply tilt the panel up, rotating it around the lower bushings.



When it's almost straight up, you'll have to lift it slightly to fit the top slots in the back of the panel over the upper bushings. Once you have it aligned, push the panel back onto the bushings, then lower the panel into the seated position.



To fold it back down for service access, lift the panel straight up to get the top slots aligned with the upper bushings again, then ease the top edge forward to get it free of the upper bushings. It'll now rotate on the lower bushings to fold down flat against the top of the cabinet.

Installing the speaker retainer rings

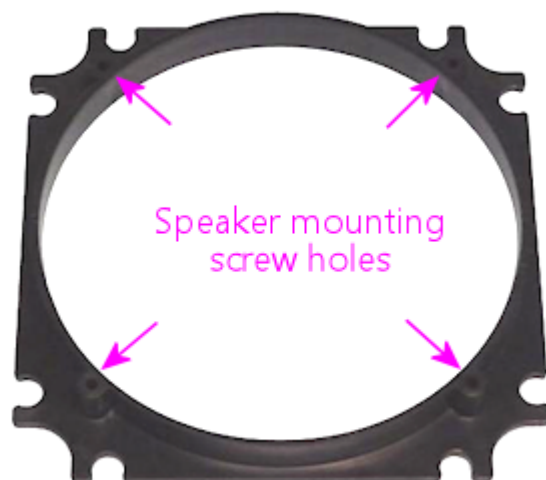
Important: if you're not using 5.25" car speakers in your panel, you might want to skip installing the retainer rings, if they're not already installed. The retainer rings are designed specifically for 5.25" speakers. For other sizes, it's better to skip the original rings and create your own adapter instead. If you *are* using 5.25" speakers, you should install the rings, since they make it really easy to mount the speakers.

In keeping with the molded plastic design, the WPC-95 panel has some companion parts called "speaker retainer rings" that are needed to install speakers.

There are two types of these retainer rings:

- "Large", for 5.25" car speakers, Williams/Bally part number 04-10382-7-2
- "Small", for 3" car speakers, Williams/Bally 04-10382-7-3

For a virtual pinball project, you'll usually want two of the 5.25" rings, which look like this:

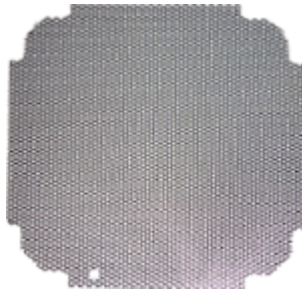


And when installed:





In addition, you'll need speaker grills that are cut into the right shape to fit the openings. Pre-cut grills are available under Williams/Bally part number 04-10382-7-4.



WPC-95 speaker grills, part #04-10382-7-4

On the 1990s machines, Williams always used an asymmetrical speaker pair, with a large midrange speaker on the left and a small tweeter on the right. You'd think this would have been a stereo pair, but it wasn't; Williams thought of the panel as a single-channel speaker with two drivers. As a result, if you order a complete WPC-95 speaker panel assembly with all of the goodies included, it'll probably include one large and one small retainer ring. So you might need to order a second large ring to accommodate your second speaker.

Some panels come with speaker rings pre-installed, and if so, the right side will usually have the small 3" ring. In this case, you'll probably want to remove the small ring so that you can replace it with a 5.25" retainer ring instead. To do so, remove the eight push nuts around the perimeter; once they're gone, you can simply lift the retainer ring off the posts. The push nuts are designed to be installed once and then never removed, so you might have to break them in half to get them out. Many people who have done this say that the best or maybe only way to extract them is to cut them in half with wire cutters; others report success prying them out with small screwdrivers. In either case, be very careful not to break the molded plastic posts while you're working. The push nuts are expendable, but the posts would be hard to fix.

If your panel didn't come with the retainer rings already installed, or if you have to install a second large ring after removing a small one on the right speaker cutout, installation is easy:

- Install the speaker grill first - it has to go under the ring. Fit it over the opening with the notches at the corners aligned with the posts in the panel.
- Fit the ring onto the eight posts around the perimeter of the speaker cutout.
- Install a 1/4"-shaft push nut on each post to secure the ring. The push nuts work just like the name says - push them onto the posts as far as possible.
- If your assembly didn't come with push nuts, or you had to destroy your old

ones to remove them, you just need some generic 1/4"-shaft push nuts. The pinball suppliers don't seem to have anything matching, so you'll have to get these from a hardware store or online. They're kind of obscure for the likes of Home Depot, so it'll probably have to be online. One part that reportedly fits is Tinnerman PS250385.

Installing 5.25" speakers

If you haven't already installed a 5.25" speaker retainer ring in each speaker cutout, follow the procedure above. If your speaker panel came with the retainer rings pre-installed, but one of them is for a small 3" speaker, you'll have to remove the 3" retainer ring and replace it with the 5.25" version. This is also explained in the section above.

Once the retainer rings are installed, there's nothing to installing 5.25" speakers. The rings have screw holes that should line up with the mounting screw holes on your speakers. Line up the speaker and fasten it with #6-32 machine screws. The length of screw needed will vary depending on the thickness of your speaker's mounting frame base. Start with 3/4" screws, and use something longer if necessary.

Installing other speaker sizes

Williams designed these panels for 5.25" and 3" speakers, period. They didn't make any attempt to facilitate installing other sizes of speakers.

If I had some other size of speaker that I wanted to install, my first inclination would be to return them and get some standard 5.25" speakers instead. It's easy to find good speakers in that size, and the integrated mounting hardware on the panel makes it practically no work to mount them (see the section above).

If you really insist on using a different speaker size, it can be done - you just have to create your own improvised adapter plate. This will take the place of the normal retainer ring.

- First, remove any existing retainer rings already installed on the panel. See the section on installing the retainer rings above for details on how to remove rings that are already in place.
- Create an adapter from a 4½" square piece of ½" thick MDF. Cut a circular opening in the middle of the MDF matching your speaker's aperture, and drill holes around the perimeter that line up with the mounting holes on the speaker. Also drill holes at the corners that match the pattern of the eight posts on the back of the speaker panel that are used to attach the speaker retainer ring.
- Mount the speaker to the MDF board using machine screws and nuts.
- Attach the MDF board to the back of the speaker panel, where the speaker retainer ring would normally go. Fasten with push nuts.
- For more detailed instructions, try Joseph "Tony" Dziedzic's tutorial for upgrading original WPC-95 speakers:

www.dziedzic.us/wpc_speaker_plastic_display_panel.html

Lighting the speakers

Some people install LED strips around the perimeter of the backbox speakers, just

for the sake of another lighting effect. See "Installing speaker LED strips" in Chapter 32, Original WPC Speaker Panel. The same approach should work for the WPC-95 speaker panel. The retainer rings should serve as a good mounting surface, assuming your speakers don't protrude too far in front of their mounting base; if they do, you can add some spacers between the speaker and retaining ring to create more room for the LED strips.

Installing a real DMD

Before installing a DMD, you should install the clear acrylic display shield (Williams/Bally part number 01-13636). Fit it over the posts and fasten with push nuts. The same push nuts used with the speaker retainer rings (such as Tinnerman PS250385) will work here.

The speaker panel is designed specifically to fit the plasma DMD devices that were used in the original WPC machines. The DMD should have mounting holes at its corners that align with screw holes in the back of the speaker panel. Align the device and fasten with #6-32 machine screws. You might need nylon spacers between the device and the panel.

Installing a video panel for the DMD

You'll probably want to install the clear acrylic display shield (Williams/Bally part number 01-13636) before installing a video display. Fit it over the posts and fasten with push nuts. The same push nuts used for the speaker retainer rings (such as Tinnerman PS250385) will work here.

The speaker panel's mounting holes are specifically designed for the plasma DMD devices used in the WPC machines. Williams didn't anticipate that these would ever be switched to a different type of display, so they made no provision for attaching modern video panels. You'll have to improvise your own brackets. Laptop panels are small enough and light enough that you don't need anything particularly strong, so this isn't too difficult, but I don't have any "standard" solution to suggest. On my "original style" speaker panel, I used sheet metal to fashion some brackets that fit over the edges of the video panel, and then fastened those to the speaker panel using the screws intended for the DMD. If you want something more off-the-shelf, you might try looking at picture/mirror hangers at a hardware store - something like a mirror clip might work.

Note that a laptop video panel in the appropriate width for the DMD window is usually just a little too tall for the speaker panel. It'll probably stick up above the top of the speaker panel by a little under an inch. That's usually not a problem, since the extra height will be hidden behind the translite.

34. Cabinet Buttons

One of the top ten questions that every new cab builder asks on the forums is:

"Which buttons should I include?"

I think most of us start off thinking the answer is so obvious that it's not even worth asking the question: why, the same buttons they put on *real* pinball machines, of course! But as soon as you start actually designing your cab, it becomes apparent that it's not as simple as it seems. Some of the complications:

- Real machines don't all have the same buttons. There are three buttons that are pretty much universal: the two flipper buttons and the Start button. Beyond that, different machines have their own extra buttons, some common and some unique. Some common buttons that appear on many machines are the Magna Save buttons (alongside the flipper buttons), a flipper-like button on top of the lock bar (often labeled "Fire"), an Extra Ball or "Buy In" button on the front of the cabinet, and a Launch Ball button in place of the plunger. If you want to be able to play re-creations of tables with those extra buttons, you'll need some equivalent of the extra buttons on your cab.
- Modern pinball machines have some additional, hidden buttons, inside the coin door, that let the operator access the game's setup menus. It's useful to have these on a virtual cab, too, because you're going to be the operator and will want to be able to access the setup menus.
- Virtual pinball needs some additional controls for the "virtual" aspect, such as inserting virtual coins, navigating the game selection menus, viewing game instruction cards, viewing flyers, exiting an active game and returning to the game selection menu, and controlling the PC audio volume. Many of these functions can be comfortably assigned to the regular pinball buttons (flippers, Start, etc), so you don't actually need a big array of extra controls, but most people find it necessary to have at least one extra button, for the special function unique to virtual pinball of exiting out of the current table.

This section will try to help you answer that big question about which buttons to include. I don't want to presume to dictate a single slate of buttons that everyone should use. Tastes vary about whether you should have lots of buttons or as few as possible, and some cabs have more space than others available for buttons. So instead of trying to answer the question with a list of "these are the buttons you should include", we'll go over all of the buttons commonly used on virtual cabs, even the ones only rarely seen, and we'll explain each one's purpose and offer an opinion on each one's importance. (There is, however, a summary of recommendations below, in case you want to cut to the case.)

Later in the section, after surveying all of the button functions, we'll get into the details of the equipment to use for each type of button, including what to buy and how to install it.

Minimalist vs. maximalist

When you add up all of the buttons used across different real pinball machines, plus all of the special buttons for virtual pinball needs, you get a pretty long list.

One way to cover all the bases is to simply include every button you can think of. It's impractical to include every unique button on every machine - there are just too many of those - but you can include all of the special buttons for all of the games you think you might want to play. This "maximalist" approach gives you the best chance of replicating the original playing experience for all of the different games. It

also gives you lots of blinking lights, and we all like shiny things.

Some cab builders prefer a more "minimalist" approach, with the aim of replicating the appearance of a real pinball machine more closely. Most real pinballs have only a few buttons, so a cabinet with too many extra buttons can start to look conspicuously "virtual". The minimalist tries to limit exterior buttons to about the same number found on most real machine, for a more authentic appearance.

Fortunately for the minimalists, the modern software environment makes it possible to access all of the important functions with only a few extra buttons beyond the core set found on all real pinballs.

If you're a determined maximalist, note that the practical limits of available space might cramp your style a bit. A standard cabinet with a coin door can really only fit a maximum of four buttons on the front panel, in addition to the plunger, the Launch Ball button, and the coin door controls.

I personally favor a middle path, closer to the minimalist side, but allowing a few extra buttons for the sake of the broadest playability. With this in mind, the list below has my recommendations for which buttons to include and which to skip, or at least find a way to hide.

Master list of buttons

To help you decide which buttons you need, whichever camp you're in, here's a list of all of the common and less common virtual cab buttons, how important they are (in my opinion), and where they're usually situated. The list is roughly in order of importance, from most important to least.

- **Power button:** *Essential.* Pushbutton, bottom of cab, front right side.
On the real machines, the power is controlled with a toggle switch wired to the main power supply line, located in a little recess on the bottom of the cab at the front right corner. In a virtual cab, we usually replace this with a "soft" power button in the same location, wired to the PC motherboard's power control header. See "Floor" in Chapter 21, Cabinet Body for details on installing a pushbutton in the proper spot, and Chapter 11, Power Switching for more on setting up the PC soft power controls.
- **Start:** *Essential.* Round pushbutton, front panel, top left.
This is a core button found on every real pinball. Push it to start the game. Most "front ends" (the game menu software that lets you select games to play) also use this as a "select" button when navigating through menus.
- **Flippers:** *Essential.* Standard flipper buttons, sides of cab.
These are core buttons found on every real pinball.
- **Exit:** *Essential.* Round pushbutton, front panel, below Start.
You won't find this button on real pinballs; you "exit" a real pinball game by walking away. But it's a must for a virtual cab. This button exits the current game and returns to the "front end" game selection menu. Front ends typically also use this to navigate within their menu structure (to back out of a menu or acknowledge a pop-up message, for example).
- **Coin In:** *Essential.* No standard location, but commonly either a round pushbutton on the front panel, or repurposing the Coin Reject buttons on the coin door slots.
This button lets you simulate inserting a coin to add credits to the current game. Even though you'll probably be setting most



of your games on Free Play via the operator menus, a Coin button is still needed in some cases, since some older tables simply don't have a Free Play option.

If you have a full coin door with coin slots, you don't absolutely need a separate button for this, since you can use actual coins instead, just like in a real arcade machine. Most coin doors don't come with coin mechs installed, so you'll probably have to separately purchase the mechs (about \$10 each for US quarter acceptors) and install them yourself. You'll also have to wire the coin slot microswitches to your key encoder, which is just like wiring a button switch. This is all covered in detail in Chapter 40, Coin Door.

Even if you do install and wire physical coin acceptors, you might still want to add a separate Coin In button, since a lot of people find it tedious to feed in quarters once the novelty wears off. The most obvious way to add a Coin In function is to simply add one more round pushbutton to the front panel. If you don't like the added visual clutter of another button, though, there's a neat way to hide it in plain sight. Specifically, you can use the "Reject Coin" buttons in your coin slots as your virtual coin buttons. To do that, you have to add microswitches behind the buttons, as explained in Chapter 40, Coin Door.

My favorite option is to include *both* the real coin mechanisms *and* the Reject button switches. That way you can use real coins when you're in the mood for the full arcade experience, and just push the buttons when you're not.

- **Magna Save:** *Essential-ish*. Standard flipper buttons, sides of cab.

This is a second set of buttons on the sides of the cabinet, just behind or below the flipper buttons, using the same button style as the flipper buttons. Everyone calls these "Magna Save" buttons, because a few games in the 1980s famously used such buttons for a magnetic ball-save feature with that trademarked name. Calling them Magna Save buttons is a little misleading, because similar buttons were used on many other pinball machines over the years for all sorts of different special features, but we really don't have a better name, so Magna Save it is.

I'm grading these "Essential-ish", because you can get by without them in the majority of games. But if you omit them, you'll give up some important features in a lot of games - many more games than you'd think based on the Magna Save name. Dozens of real tables featured similar buttons. Even if you're a hard-core minimalist about buttons on your cab, there's a case to be made that these buttons appear on enough real pinball machines to be considered part of the basic set of authentic pinball controls. But the real reason to include them is that an even greater number of Visual Pinball tables rely on them for special game-specific features, because these buttons have become the standard fallback for emulating unique controls. You really need these buttons on your cab if you want to enjoy full playability for many Visual Pinball tables. To see what I mean, take a look at the lists in Appendix 4, Tables with MagnaSave Buttons.

- **Tilt bob:** *Essential-ish*. Inside the cab.

The tilt bob is a little mechanical pendulum that detects excess cabinet motion. This isn't exactly a button, but it acts like one in almost every way - you connect it to the key encoder like a button, and it provides input to the software as though it were a button.

Don't confuse the tilt bob with a nudge sensor. The purpose of a tilt bob isn't to simulate nudges in the software, but rather to detect when you're nudging *too much*. The tilt bob serves exactly the same function



in a virtual cab that it serves in a real pinball machine. See Chapter 36, Nudge & Tilt for a full discussion of the distinction between "nudging" and "tilting", and the tilt bob's role in this.

Visual Pinball has its own simulated tilt bob, but I strongly recommend installing a physical tilt bob anyway. VP's version just isn't very realistic. The interaction between a real cabinet's motion and the tilt bob's motion is complex; it's one of those compound pendulums that's rather difficult to model, and VP doesn't really try. And I don't see why it should, when we can so easily get *exactly* the right handling by just hooking up a real tilt bob! That's why I rank this as practically essential. And it doesn't affect aesthetics, since it goes inside the cab.

- **Launch Ball:** *Nice to have if you have a plunger; essential if not.*
Large round pushbutton, front panel, right.

A fair number of games from the 1990s used a Launch Ball button in place of a regular plunger. This was usually a large round pushbutton, but some games used unique devices instead, such as a gun trigger on *Terminator 2: Judgment Day* or a gear shift lever on *The Getaway: High Speed 2*. On the real machines that use such a thing, the button or device is typically positioned exactly where the plunger would normally go, at the top right corner of the front panel.



For a virtual cab, this button is optional if you're going to install a plunger using the Pinscape Controller or the plunger kit from Zeb's Boards. Both of those let you simulate the Launch Ball button using the plunger, so you don't absolutely need the button. Even so, I still recommend including the button in addition to the plunger, since it replicates the original playing experience more faithfully for Launch Ball tables.

Alternatively, you can use a Launch button *instead of* a regular plunger. All of the PC pinball simulators are designed with desktop use in mind, so they all let you control the on-screen plunger with a key press. You can simply map the Launch Ball button to the software plunger key and dispense with the real plunger. This saves front panel space and the complexity and cost of a real plunger.

In my opinion, the best option is to have *both* a regular plunger and a Launch Ball button. Controlling the on-screen plunger with a button is possible, but it's a really poor substitute for a genuine plunger. Besides, the plunger is a defining feature of pinball; for me it's simply a must in a virtual cab. If you do choose to include both the plunger and the button, the button can go just above or just below the plunger. I prefer to put the plunger in its standard location on a real pinball, which means the button has to go below it. But some people prefer putting the button above the plunger to make it easier to see and reach, which requires lowering the plunger by a couple of inches from the standard position to make room for the button. I'm not wild about the unusual appearance of the lowered plunger, but some cab builders have to lower the plunger anyway because the normal position is blocked by the playfield TV.

- **Audio volume:** *Recommended.* No standard location; preferably hidden.
I find that I adjust the audio volume quite frequently on my cab, so I very much like have a way to do this conveniently. Getting out the keyboard or mouse or opening up the cab don't count as convenient, so some kind of

external control for this is recommended.

Some people like to use the straightforward approach of adjusting the volume via the amplifier's volume knob, so they locate the amp somewhere easily accessible.

My preference is to control the audio volume via external cabinet buttons for Volume Up and Volume Down. This is possible if you use some software to provide a keyboard interface to the Windows master line-out volume level, such as my PinVol utility.

If your keyboard encoder has the concept of a "Shift" button (Pinscape and the i-Pac encoders do), you don't need separate physical buttons for volume. You can instead double up some of the regular buttons, using them as volume controls in addition to their normal functions. This is the approach I use myself, and I like it because it makes the volume controls easily accessible without requiring any extra physical buttons. I use the right MagnaSave/flipper buttons as Volume Up/Down buttons, in combination with my Extra Ball button as the Shift button. That is, when I press and hold Extra Ball, I can use the right buttons to adjust the volume. Easy, convenient, and requires no extra buttons.

If you prefer separate, physical Volume Up/Down buttons, where do they go? There's no standard location for these, but even the maximalists seem to agree that these particular buttons should be hidden. Some cab builders accomplish this by putting them inside the coin door. That's a little too inconvenient, in my opinion, for something I access so often. A better option is the bottom of the cabinet, near the front edge: that puts them out of sight, but they're easily accessible and you can find them by feel.

Another option that you might prefer is a knob that sends volume up/down keystrokes to the PC. See the vpfforums thread Coin Door Volume Control for a way to build such a knob and mount it on the coin door.

- **Coin door position switch:** *Recommended.* Inside the coin door.

The real machines have a switch inside the coin door that senses whether the door is open or closed, like the switch in a refrigerator that turns on the light when you open the door. I'm sure this seems like a ridiculously nitpicky detail, but it's actually important, because a lot of the ROM software on the modern games requires the signal from this switch to access the service menus. (They're so insistent about it because they use it as a safety measure, to cut off the high voltage power supply when the operator is working inside the machine.) Installing the switch to match the software's expectations just makes things easier, and it's not difficult or expensive to install. For full details, see "Coin door position switch" in Chapter 40, Coin Door.

- **Fire!:** *Nice to have; essential for Stern fans.* Top center of lockdown bar.

Many newer Stern games (2005 to present) feature this extra button on top of the lockdown bar, usually centered side to side. (The lockdown bar is the metal piece at the front edge holding the cover glass down.) The button itself is usually a clear flipper button and usually has a lamp inside.



Up until recently, I think many cab builders considered the Fire button to be one of those special, game-specific buttons that isn't widely enough used to justify including it as a physical button. But it's not *that* game-specific any more, given that nearly every newer Stern title has one. I think it's on its way to becoming a

new core button on the strength of Stern's consistent use, and because it's just a good location for an extra button. It's visually appealing even if you're in the minimalist camp, and it's easy to reach during play.

The complication is that you either need a special (more expensive) version of both the lockbar and receiver that are pre-drilled to accept the button, or you need to modify the standard ones. See "Fire button" in Chapter 23, Cabinet Hardware Installation for more on finding the required parts.

- **Service buttons:** *Nice to have.* Inside the coin door.

If you're using a real coin door, it will probably come with a little "service button" panel pre-installed on the inside of the door. Modern coin doors for WPC and Stern machines have four buttons here; older 1980s games usually had three buttons. On a real machine, these let the operator access the setup menus for the game, which are used adjust pricing and feature settings and to access self-diagnostics. These menus are where you set up Free Play, for example, or change from 3-ball to 5-ball play. In a virtual cab, you can access the same menus via the authentic buttons if you have them, by connecting the buttons to your key encoder.



If you don't have a real coin door, or it doesn't include the service buttons, you can access the service menus via your PC keyboard. This isn't something you have to do very often, so I don't consider these buttons to be essential. But I do consider them to be a big convenience that I wouldn't want to do without on my own machine. Assuming you have a coin door, these buttons don't affect the aesthetics, since they're hidden inside the machine.

- **Extra Ball/Buy-In:** *Nice to have.* Round pushbutton, below Start.

A few games from the 1990s feature an extra front-panel button labeled "Extra Ball", "Buy-In", or in a few cases something marketing-ese like "Super Ball". The button on the real machines is either below the Start button or below the plunger. It lets you buy an extra ball for a credit after the last ball ends (which can be a decent deal if you're close to a replay or close to beating the high score). Realistically, for virtual play, you'll probably never use it: why bother "buying" an extra ball when every game is free? Even so, I included one on my machine mostly for completeness, because I like a lot of games from the era where this button was common. If you do include it, it also gives you one more option for mapping special functions in games with unique buttons.



- **Night Mode:** *Nice to have.* Hidden; no standard location.

If your cabinet has mechanical feedback devices (contactors, solenoids, replay knocker, etc), you might want to include a button or switch to activate "Night Mode", which turns off the noisy devices for quieter play in the wee hours. The wiring depends on your output device controller; if you're using a Pinscape Controller for feedback devices, you can assign any button input to serve as the Night Mode button. The generic arcade controllers (LedWiz, PacLed) don't have this feature at all, but you can improvise something with a switch that cuts off power to the noisy devices. See Chapter 44, Feedback Devices Overview. Like the audio volume controls, this button is an "operator" button that you'll probably want to place somewhere out of view, such as on the bottom of the machine or inside the coin door. With Pinscape, you can assign it as a secondary ("shifted") function of one of your other buttons.

- **Slam tilt switch:** Neutral. If you have a WPC-style coin door, it will probably have a slam tilt switch pre-installed. The slam tilt looks like a leaf switch with a

big metal slug attached to the end of the main leaf. It's usually mounted on the inside of the coin door.



This is related to the tilt bob, but rather than detecting excess motion, it detects excess force. It's triggered by very hard jolts to the front of the cabinet. That big round slug at the end of the main leaf has a lot of inertia, which is what makes the switch work. A hard enough slam will propel the slug by sheer inertia far enough to close the switch and trigger the alarm.

I don't find the slam tilt to be all that necessary on a virtual cab, since its main purpose is to deter the sort of extreme abuse that a coin-op machine in a public place can be subjected to. It's unlikely you'd ever trigger it in regular play, even if you're an aggressive nudger. So I wouldn't go out of my way to add this device if your coin door doesn't already have one. But if it's already there, I'd go ahead and wire it up, since VPinMAME does recognize it as a standard input. The WPC doors usually do come with a slam tilt switch; the Stern and SuzoHapp doors don't.

See Chapter 40, Coin Door for more on wiring a pre-installed slam tilt switch and information about how it's treated in the software.

- **Virtual service buttons:** *Kinda nice to have.* Inside the coin door. There are a few miscellaneous "operator" functions specific to virtual pinball that might merit their own buttons inside the coin door, such as engaging night mode, doing a hard reboot of the PC or activating plunger calibration mode. One way to do this is to add your own bank of additional service buttons somewhere inside the coin door. See "Adding an extra service panel" in Chapter 40, Coin Door for ideas.
- **Flyer, Instructions, Info:** *For the maximalist only.* Below Start. Some front ends (game selection menu programs) use keys to select selection certain functions within the UI, such as viewing a game's promotional flyer, viewing its instruction card, or displaying other information about the game. Some cab builders include a bunch of cabinet buttons dedicated to these front-end functions. Functionally, you don't really need a bunch of special buttons for the menu program, since the newer front ends let you access everything through the basic keys. You can include them if you like the aesthetics of lots of buttons, though, and they do give you more options for mapping special functions in tables that had unique extra buttons of their own.
- **Nudge:** *Not recommended.* No standard location. In the early days of virtual cabs, some cab builders added special extra buttons for nudging. This simply transplanted one of the bad features of desktop play (the lack of real physical interaction) into the cabinet, which is the opposite of our ultimate goal of emulating the elements of real pinball play in software. But it was the only option back then, so people did it to have some kind of nudge interaction, even if it wasn't very good. Fortunately, there's a much better way to handle nudging now: use an accelerometer. That lets you *virtually* nudge the simulated game by *actually* nudging the physical cabinet. Unlike the bad old days of push-button nudging, where the nudge command was one-size-fits-all, the accelerometer registers the amount and direction of the force you apply, so that the software response can be proportional. It makes the play much more natural and immersive than trying to fumble for buttons at critical moments. See Chapter 36, Nudge & Tilt.

Recommendations

Here's a summary of my recommendations, in descending order of importance.

Essential (all cabs need these):

- Power button
- Start
- Flippers
- Exit
- Coin In

All but essential:

- Magna Save
- Tilt bob

Recommended:

- Audio volume up/down/mute
- Coin door position switch

Nice to have:

- Launch ball (essential if you don't have a plunger)
- Service buttons
- Extra Ball/Buy-In
- Night Mode
- Stern-style lockbar Fire button

Neutral:

- Slam Tilt
- Virtual service buttons

For the maximalist only:

- Flyer
- Instructions
- Info

Not recommended:

- Nudge buttons (use an accelerometer instead!)

Illuminated buttons

Most people want illuminated buttons for the common front panel buttons - Start, Exit, Launch Ball, Extra Ball, Coin In, etc.

This is fortunately really easy to do, because you can buy pre-assembled button-plus-lamp assemblies in the "small round pushbutton" style used on modern pinball machines.

Most people further want to have the software control the button lights, so that the Start button flashes when the game is ready to start and the Launch Ball button flashes when a ball is ready to launch, and so on. This is also easily done, as long as

you have a feedback controller like an LedWiz or Pinscape Controller.

A lot of first-time cab builders find this idea little confusing at first. The big question a lot of people ask is how you're supposed to connect a button lamp to your keyboard encoder. The answer is that you're *not* supposed to connect the lamp to your encoder. You're supposed to connect it to your feedback controller.

This becomes a lot easier to grasp if you think about the *switch* and the *lamp* as two separate devices. What throws people is that the two are built into a single plastic housing. But, electronically, they're not connected at all. Think about it like this:

- there's a switch, which you connect to your keyboard encoder
- and there's a lamp, which you connect to your feedback controller

But then you might wonder: How does the feedback controller know that the lamp goes with the button, and how does the keyboard encoder know that the switch goes with the lamp? Well, they don't. And they don't have to. The key input and the feedback output port are separate entities in the software as well. So you don't have to worry about coordinating between the two physical devices in terms of the wiring; it's all handled in the software.

The switch wiring part is explained in more detail in Chapter 35, Button Wiring.

The lamp wiring part is covered in Chapter 55, Button Lamps.

Incandescent and LED lamps

Most of the button-with-lamp assemblies that you can buy from the pinball vendors come with incandescent lamps. The standard lamp type for most of these is called a #555 bulb. These are longish cylindrical bulbs with wedge-shaped bases. They run on 6.3V power and draw about 250mA of current.



Plug-and-play LED replacement bulbs are available for the #555 incandescents. You can find these at the pinball vendors as well as from Amazon or eBay. The LED versions draw considerably less power than the incandescent versions and run at full brightness on 5V (the incandescents need the unusual voltage of 6.3V for full brightness).

If you do opt for LED replacements, pay attention to color. White LEDs are actually composed of separate red/green/blue elements, so if you put a white LED behind a blue-tinted button cover, two-thirds of the light (the red and green parts) will be blocked out, so the light will only look 1/3 as bright as it should. What you should do instead is match the color of the LED to the button color as closely as you can: for a yellow button face, buy a yellow LED.

Regular vs. "non-ghosting" LEDs: Many of the pinball vendors offer two types of LED replacement bulbs: regular and "non-ghosting". The non-ghosting kind is marketed as a premium upgrade, so at first glance you might think it's worth spending a few extra dollars to get the "best". But beware! For virtual cab use, the non-ghosting type are actually a **downgrade**. A non-ghosting LED has a little capacitor inside that makes it switch on and off slowly. This is specifically designed to work around a strobing problem that can occur if you plug regular LEDs into older pinball machines. That strobing effect was a result of the design of the older machines, and it **doesn't** happen with the modern PWM controllers we use in virtual

pin cabs. In fact, the delayed switching is actively harmful in pin cabs, because it interferes with PWM brightness control. So you should **avoid non-ghosting LEDs** for your virtual pin cab - use the regular kind instead. In most cases, if the vendor doesn't specifically say they're non-ghosting, they're probably not, because non-ghosting is inherently more expensive and so the seller will want to call attention to it as a selling point.

(An aside, for those interested in the technical details of the strobing effect. The pinball machines of the 1980s and 1990s used what they called a "lamp matrix" to operate the controlled lamps - the lamps that the controller could switch on and off individually during the game, such as an "Extra Ball When Lit" light. The lamp matrix was a way of reducing the amount of wiring and the number of transistors needed in the main control board. Rather than dedicating a transistor switching circuit to each lamp individually, the lamp matrix placed each lamp at the intersection of a "row" and "column" in the matrix, so that they only needed one transistor per row and one per column. This let them control 64 lamps with 8 row transistors and 8 column transistors. Transistors were more expensive in those days, so controlling 64 lamps with 16 transistors was a significant cost savings. Because of the shared transistors, though, the controller could only turn on one column at a time, so it had to rapidly cycle through the columns to make all of the lamps light up. This worked fine with incandescent bulbs, because incandescent filaments take a fairly long time to heat up and cool down. Even though each lamp was energized only briefly as the controller cycled through the columns, the filaments stayed hot enough between pulses that the bulbs kept glowing, with only a slight dip in brightness between pulses, too slight for the eye to see. When people started replacing the bulbs with LEDs, the column cycling became visible, because LEDs turn on and off almost instantly. They don't have filaments that retain heat between pulses. The column strobing was slow enough that the human eye could easily see it with an LED. So that's where non-ghosting LEDs come in. They have capacitors inside that replicate the slow switching times of incandescents, so that each lamp stays on long enough that it doesn't flash between column cycles. This is actively harmful with modern PWM controllers, because PWM also uses rapid flashing, to control brightness. In the case of PWM, the flashing is so fast that the human eye can't see it, so we don't *need* to filter it out, and we don't *want* to filter it out, since that would cancel out the brightness controls. If you put in the non-ghosting capacitors, the PWM controller won't be able to create the brightness fading effects that we want.)

Common pinball button types

Small round pushbutton

Most cab builders use the SuzoHapp "small round pushbutton" for the Start, Exit, and (if present) Extra Ball buttons. These can also be used for any other front-panel buttons you're including, such as Coin In, Instructions, Flyer, etc.



This is the same type used on most real machines made from the 1990s to present, including the WPC machines and newer Stern machine, so it looks exactly like the real article. It's convenient to install because it combines the plastic housing, a microswitch, and a lamp in a self-contained package. No other parts are needed for installation; you just pop it through a 1" diameter hole and fasten it with the included nut.

You can buy these buttons in assorted face colors, with no labels installed, directly from the manufacturer, SuzoHapp. Search for part D54-0004-2.

They're also available from pinball suppliers with various pre-installed labels used on real machines (Start, Extra Ball, Super Ball, Buy In). Search for part numbers 20-9663, 500-6388-02.

I'd buy the pre-labeled buttons to the extent you can, since they're easier, but you won't find pre-labeled buttons for virtual cab functions like Exit, Coin In, etc. You'll have to make your own labels for those. Fortunately, it's pretty easy; see "Custom labels" below.

Installation

To install with the button flush with the outside surface:

- First, route a $1\frac{3}{8}$ " diameter inset, $\frac{3}{8}$ " deep
- Then, drill the rest of the way through with a 1" hole saw bit or Forstner bit (don't use a spade bit - they cause too much chipping with plywood)



The routed inset recesses the button enough that it'll be roughly flush with the outside wall. If you want to keep it simpler and you don't mind having the button stick out a bit, you can skip the inset, and instead just drill a single hole straight through with the 1" hole saw bit or Forstner bit.

- Gently twist the microswitch base 1/8 turn to release it

Twist 1/8 turn



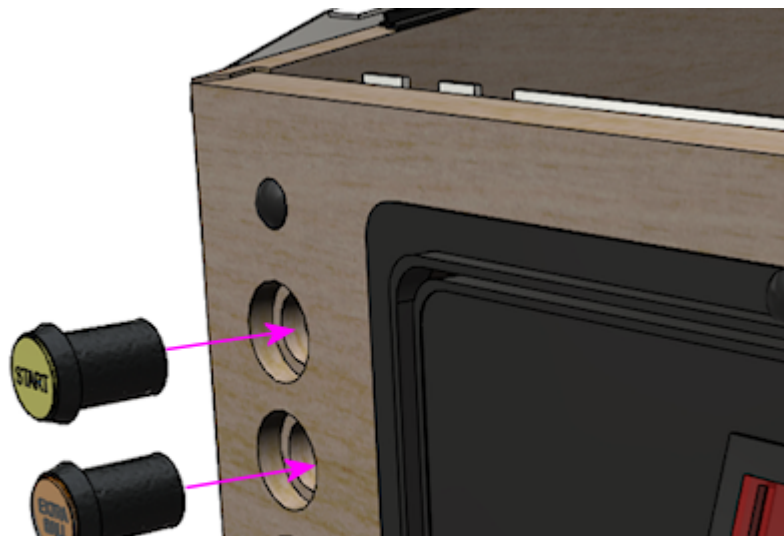
- Pull the base out



- Remove the nut



- Fit the button into the hole you drilled for it. Insert the button from the outside of the cabinet.





- Fit the nut over the shaft and tighten
- Fit the microswitch base back into the main body; when you find the spot where it fits, press it in slightly and give it a 1/8 turn to lock it back in place.

Wiring

Wiring these illuminated buttons is like wiring two completely separate devices: the switch, which you connect to your keyboard encoder; and the lamp, which you connect to your feedback device controller.

For details on wiring the switch to the key encoder, see Chapter 35, Button Wiring.

For how to wire the lamp to the feedback controller, see Chapter 55, Button Lamps.

If you don't care about controlling the lamp through software, you can just wire the lamp directly to a power supply so that it's always illuminated. Some cabinet buttons (such as Exit and Coin In) don't really have any interesting software effects anyway, so you can save a little effort this way (and conserve ports on your feedback controller). See Chapter 55, Button Lamps for how to identify the lamp terminals, then just connect those terminals directly to the appropriate power supply. #555 incandescent bulbs are designed to run on 6.3V (they'll work on 5V as well, but they'll be a little dimmer than intended). #555 LED bulbs are a little more forgiving about voltage and should look fully bright at 5V.

Typical key assignments

Here are the usual key assignments for the front-panel buttons:

- Start = "1" (the digit "1" key)
- Extra Ball = "2" (the digit "2" key)
- Coin In = "3" (the digit "3" key; "5" represents the second or right-hand coin chute in a door with two chutes with different denominations, such as dimes and quarters; "4" represents the middle chute for three-denomination doors, which are rarely seen in the US but are more common abroad; "6" represents a fourth chute, which is typically a dollar bill acceptor in the US)
- Exit = "Esc" (the Escape key)

Custom labels

You can install a custom label in these buttons by prying the lens cap off (see the step-by-step instructions below) and inserting a new label printed on transparency film.

If you want to customize the label, it's best to start with a *blank* button, rather than one that's already labeled "Start" (or whatever). The pre-labeled ones from the pinball vendors usually have the label text printed directly on the little plastic diffuser inside the lens that gives the button face its color. I don't know of a way to remove the pre-printed text. So the best bet is to buy buttons that don't have any labels in them at all.

You can buy blank buttons from SuzoHapp. They're the original manufacturer of the buttons the pinball vendors sell, so you're getting exactly the same button, directly from the source. They also offer the buttons in more colors than you can usually find

at the pinball vendors. Search on the SuzoHapp site for **D54-0004-2** to get the list of available colors.

Design your labels to fit a 7/8" circle, as illustrated below. These are just some random examples for you to use as templates for your own custom buttons, but feel free to use any of these you like.

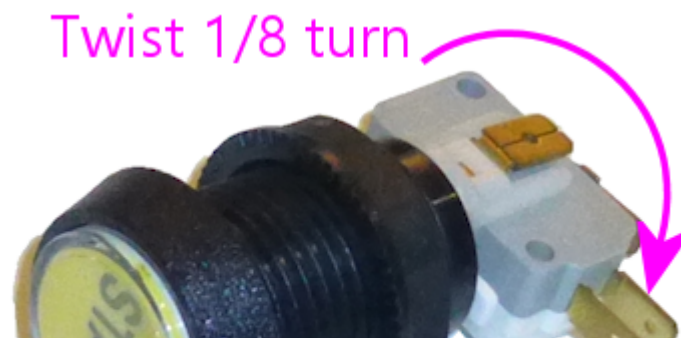


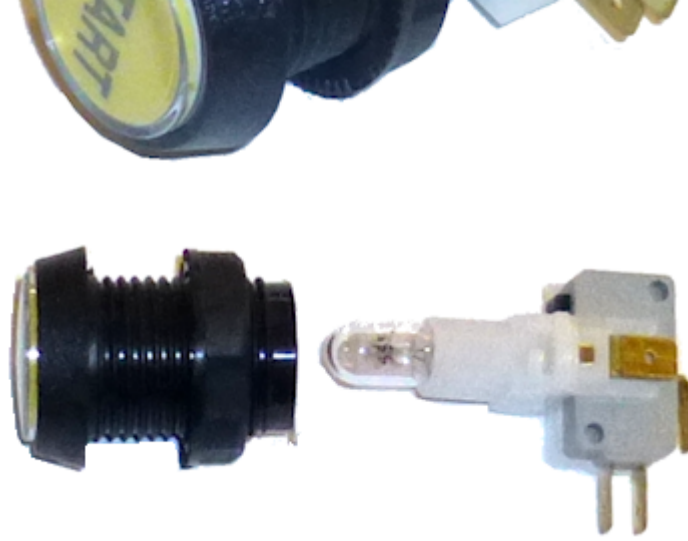
Some tips for printing:

- Use a laser printer only, not an ink-jet printer. Ink-jet inks aren't opaque enough
- Print onto transparency film
- Use film that's specifically designed for laser printers - it has a special coating that makes the laser toner stick properly
- You can use white paper in a pinch, but it's a poor substitute; the paper grain will be visible, and the text won't be opaque enough
- After printing, use an X-Acto knife to cut out the label *just inside the circle*

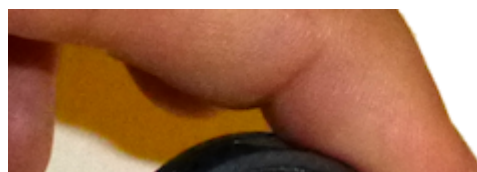
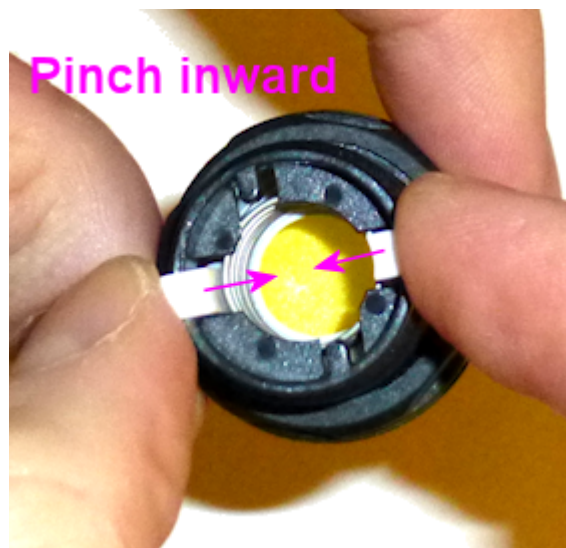
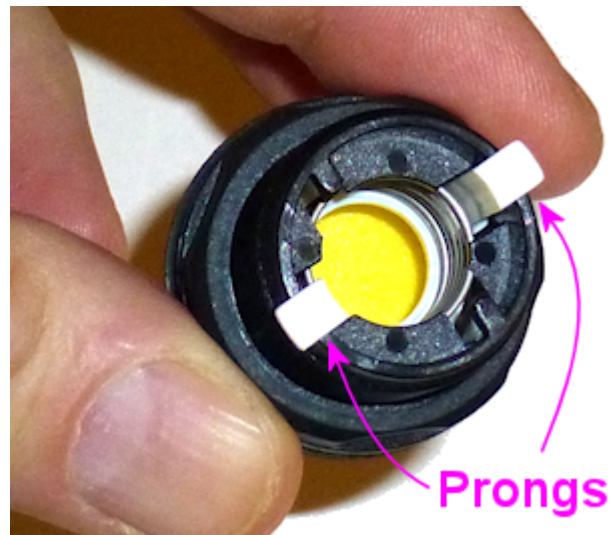
To install the label in the button:

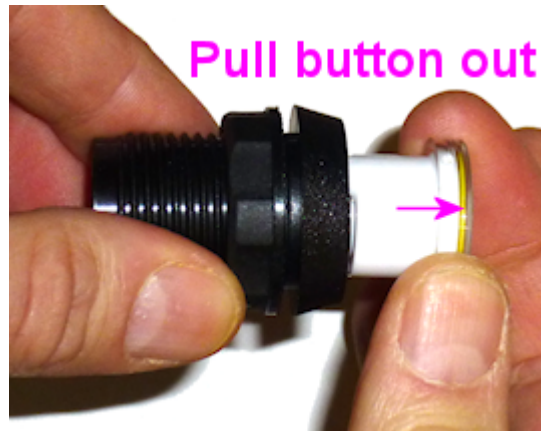
- Twist the squarish micro-switch base about 1/8 of a turn to release it, then pull it out of the housing



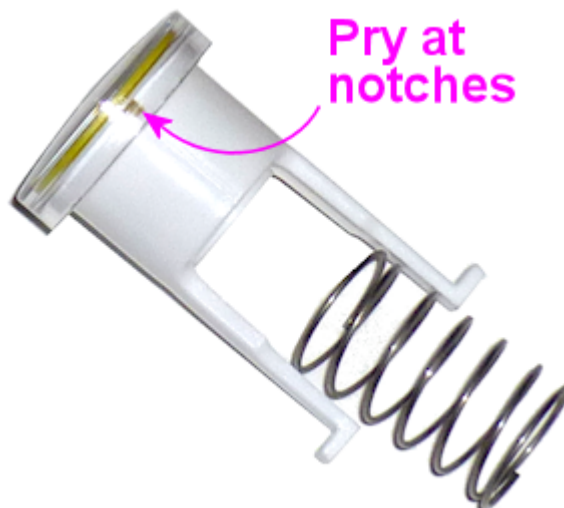


- On the bottom of the housing, pinch the white prongs together and push them into the housing. This will free the movable part of the pushbutton and let you pull it out the front.





- Now the slightly hard part: pry the lens off the top. This isn't actually that difficult, except that it'll seem at first like it's stuck on there with super-glue and won't come off without shattering. It's actually just snapped on. Use a very small screwdriver, like the type for fixing eyeglasses, and slip it under the lip. Aim for the little notches in the perimeter of the white piece underneath.





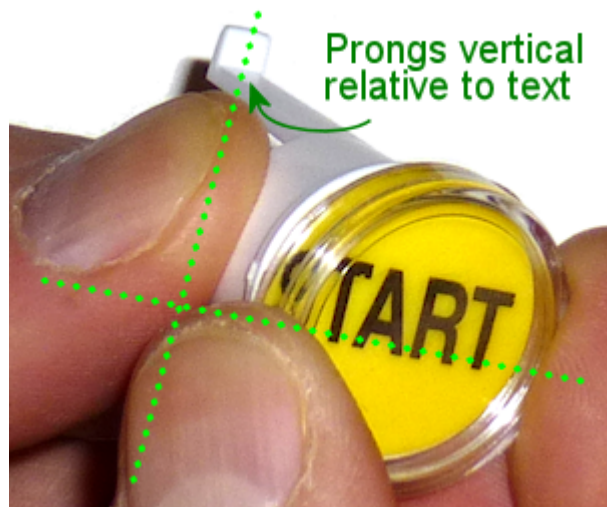
- Pry the lens a little bit at a time at quarter turns until it comes loose; once it's loose, you can just pull it off.



- You'll now have three pieces: the button body, the clear plastic lens, and the colored diffuser.



- It's easiest to get the label aligned if you start by fitting into the lens, printing side facing out through the lens.
- Fit the diffuser back over the button body, place the lens (with your new custom label) over the diffuser. Note that orientation: the text should be oriented so that the prongs are "vertical" relative to the text.



- Fit the spring back into the button body between the prongs



- Compress the spring enough that you can pinch the prongs together and fit them back into the main housing. Push the prongs all the way through the other side.



Launch Ball button

The standard Launch Ball button is the SuzoHapp "large round pushbutton" (the red one is SuzoHapp part number D54-0004-10; other colors are available with the same prefix, D54-0004-1). This is the type used on several Williams machines from the 1990s that used launch buttons in place of the standard plunger, so it's the one to use for an authentic appearance.

The SuzoHapp version is sold with a blank face with no printed label. Pinball vendors sell these with pre-printed "Launch Ball" labels in various font styles - look for Williams/Bally part number 20-9663-B-4.

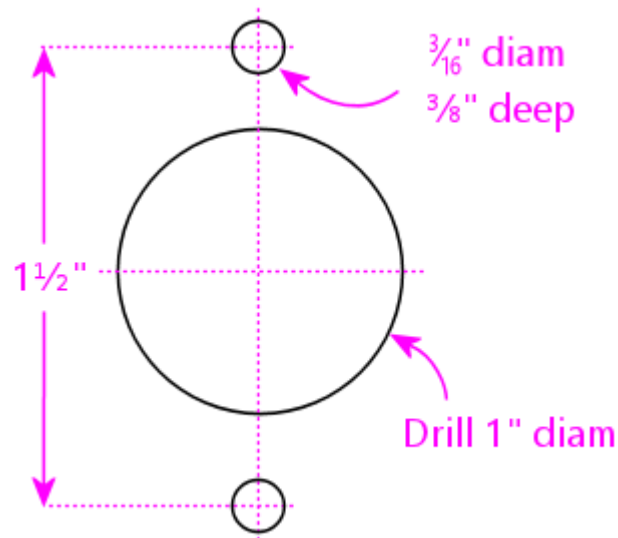


As with the small round pushbuttons, it's possible to use custom labels if you don't

like any of the pre-printed options. The process is the same as for the small buttons, as described above.

These buttons are in almost every respect exactly like the "small round pushbuttons" described above. The only real differences are that the outer button face is larger, and the drilling pattern is slightly different.

Here's the drilling pattern for these buttons. The small holes above and below the main 1" drill are for little "nubs" on the back of the button face that help align the button and prevent it from rotating freely once installed. These don't have to be drilled all the way through; the nubs are only a couple of millimeters deep.



For wiring instructions, see Chapter 35, Button Wiring and Chapter 55, Button Lamps.

The typical keyboard assignment for the Launch Ball button is the "Enter" key.

Unique ball launchers

The Big Red Button above was used on several real machines from the 1990s (*Medieval Madness*, *Attack from Mars*, *Monster Bash*, *Champion Pub*, among others), so it's a perfectly authentic choice.

But there's another option that provides an opportunity for extending your artwork theme. A few games that had auto-launchers used unique theme-based toys for the launch controls. Functionally, most of these were no different from plain buttons, but they added novelty and echoed the game's theme. Some examples:

- The gun grip launcher on *Terminator 2: Judgment Day* (Williams, 1991)
- The phaser pistol launcher on *Star Trek: The Next Generation* (Williams, 1993) (which was substantially the same equipment as *T2*'s gun launcher)
- The gun launcher on *Indiana Jones: The Pinball Adventure* (Williams, 1993) (exactly the same idea as the *ST:TNG* and *T2* launchers, but new equipment, styled to look like an antique pistol)
- The fishing reel launcher on *Fish Tales* (Williams, 1992)
- The gear shifter lever on *The Getaway: High Speed II* (Williams, 1992) (which, unlike most of the other unique launchers, is more than just a dressed-up button: it has two actions, Shift Up and Shift Down, which are used throughout the game)

Many of these unique controls are available as replacement parts from the pinball vendors, so if one of them fits your theme, you could buy one and use it in place of the generic round button.

Or, if you want to get more creative, you could design your own and fabricate it with 3D printing. Most of the unique launchers on the real machines were just fancy plastic housings with a small pushbutton embedded. You can find suitable pushbuttons from Mouser or DigiKey, and embed one in your own custom plastic housing to create theme-specific launcher toy.

Rectangular pushbuttons

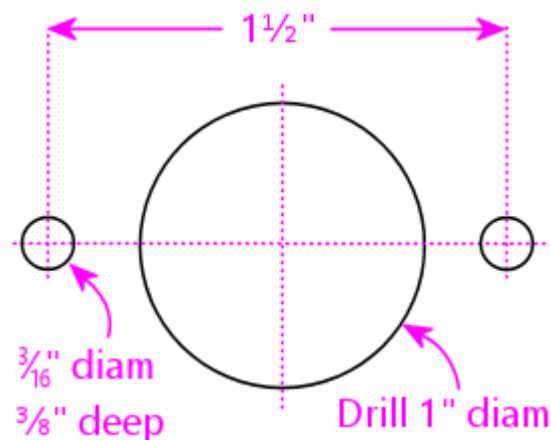
Some cab builders prefer a rectangular or square shape for some of their buttons. These are particularly popular among the "maximalists" who want the Mission Control look for the cab's front face. These also work well as hidden buttons on the bottom of a cab, since they're large enough that you can easily operate them by feel. I recommend the large square button below for the main power button, for example.

Variations of the SuzoHapp pushbuttons are available to fit these needs, such as their "large rectangular pushbutton":



Search for SuzoHapp part number prefix D54-0004-5 for this button in assorted face colors, without any pre-printed labels. You can add custom labels to these using the same procedure described for the small round pushbuttons above.

To prepare for installation for the large rectangular buttons pictured above, use a hole saw bit or Forstner bit to drill a 1" center hole. You also have to drill shallow indents, about $\frac{3}{16}$ " diameter and $\frac{3}{8}$ " deep, at either side of the button. The indents are for small "nubs" that keep the button from rotating when installed.



A variation with a square button face is also available; search for part number prefix D54-0004-4. These only require a simple 1" diameter hole for mounting.

In every other respect apart from the shape of the face, these buttons are identical to the "small round pushbuttons" detailed above. Installation and wiring are the same.

Flipper buttons

Here's the thing you have to know about flipper buttons: only leaf switches will do.

Some first-time cab builders want to use arcade buttons with built-in microswitches, like the SuzoHapp pushbuttons above or similar buttons from DIY arcade cabinet companies. You might be especially inclined in this direction if you've built a MAME video game cabinet in the past, since you've probably already bought buttons from these DIY arcade places before and it's easier to go with what you know. Even if you're not a MAME cab veteran, the assembled pushbuttons can seem attractive just because they're easier, being self-contained. And indeed, they're great for Start buttons, Exit buttons, and pretty much all of the other buttons - but *not* for flipper buttons.

The problem with the assembled buttons is that they don't have the right feel for the flippers. Flipper buttons have a very specific mechanical and electrical action that comes from the **leaf switch** inside. The microswitches used in most of the self-contained buttons have a mechanical snap action instead, which feels very different and doesn't have the same fast and precise response you get with a leaf switch.

I also think it's important to use standard flipper buttons as the pushbutton part, also to get the feel right. They have a specific size, shape, and spring weight. The arcade buttons are designed for video games; they don't try to match the mechanical feel of a flipper button.

My advice is to just do this the right way: Go to a pinball supply vendor (Pinball Life, Marco Specialties, VirtuaPin) and get a set of the real pinball flipper flippers and the matching leaf switches. Don't waste your time trying to find "close enough" products from the MAME gadget Web stores; the exact right buttons are easy to find at any of the pinball vendors.

Here's the full menu of parts:

- Flipper buttons:
 - Standard length (1-1/8"); Williams/Bally part A-16883
 - **Or:** Longer length (1-3/8"); Williams/Bally 3A-7531-5, 3A-7531-9; Stern A-711, 515-7791-00
- Pal nuts (one per button), metal or nylon; Williams/Bally 02-3000
- Leaf switches:
 - Flipper leaf switch, single contact, low voltage (gold contact points); Data East/Stern 180-5048-01
 - **Or:** Double-contact leaf switch, low voltage (gold contact points); Williams/Bally SW-1A-192
- Optional: VirtuaPin leaf switch bracket; works with longer (1-3/8") button only

You don't need all of these - for each button position, you just need one of the buttons (shorter or longer), one of the leaf switches, one Pal nut, and, if desired, the switch bracket. Read on for details of the variations and how to choose.

Which color?

If you're going to illuminate your flipper buttons (which is a nice effect - see "Flipper Buttons" in Chapter 55, Button Lamps), get them in clear transparent. That lets you light them up in any color under software control.

If you don't want lighted flipper buttons, choose whatever color you like with your cabinet artwork. You can buy transparent or opaque buttons in a wide range of colors in the standard 1-1/8" length (the longer 1-3/8" buttons are more limited in this regard). I personally prefer the appearance of the transparent ones over the opaque buttons, even when they're not illuminated, but that's just an arbitrary preference.

Longer or shorter button length?



Nearly all of the machines made from the 1980s onward used the shorter 1-1/8" flipper buttons, so I consider these the "standard" length, and they're the easiest to find and have the most color variations available.

Many older machines used the longer 1-3/8" flippers, so this longer length is still being made, but it's not in as much demand so they don't make them in as many colors. The only options I've seen are opaque red, opaque white, and clear transparent. Clear transparent is all you need if you're lighting the buttons, but if not, I suppose the limited color choices might be a reason to use the shorter ones.

Reasons to use the longer buttons:

- They work with the VirtuaPin switch brackets

Reasons to use the shorter buttons:

- They're available in more colors
- They're easier to fit into the limited space next to the plunger (especially if you're using a plunger position sensor that needs extra space)

I personally use the shorter ones, but mostly because the longer ones weren't available in clear transparent when I built my cab, and I needed transparent buttons so I could illuminate them. A transparent version of the longer button is now available now, so that's no longer a limiting factor. I might still use the shorter ones anyway, though, because they really do fit better on the plunger side.

Installing the button

For the recommended drilling locations, see the "Flipper buttons" in Chapter 21, Cabinet Body.

On the real machines, they usually drill a three-level structure for each flipper button:

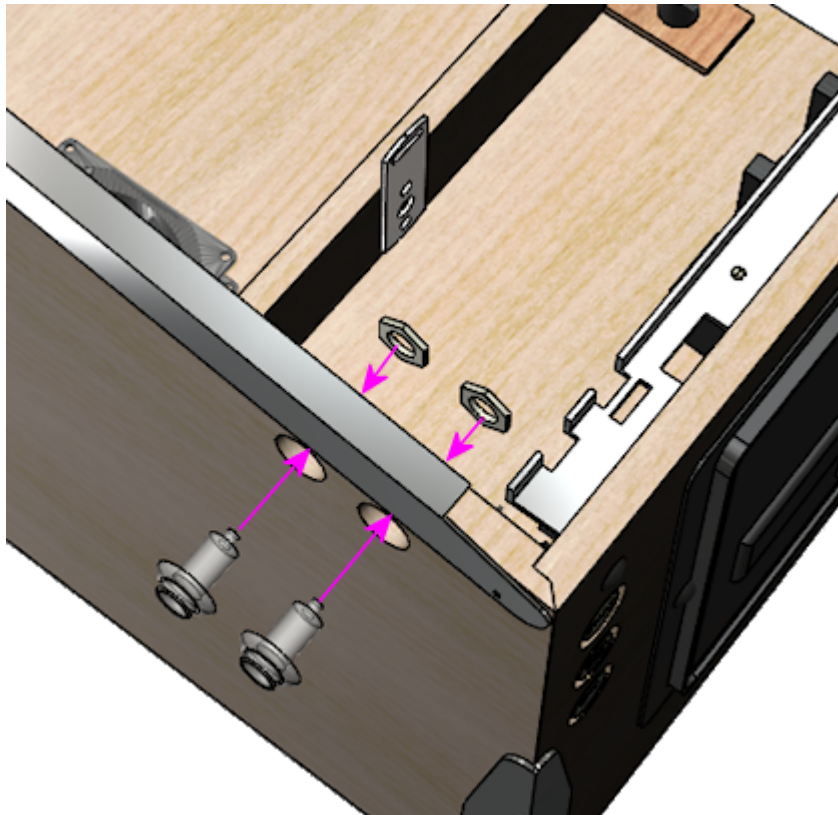
- Using a **1 1/8" diameter** hole saw, Forstner bit, or straight router bit, drill 5/16" deep from the **outside**
- Using the same **1 1/8" diameter** bit, drill 3/16" deep from the **inside**, on the same center
- Drill the rest of the way through, on the same center, with a 5/8" drill bit

This three-level structure creates a narrow "neck" that the central shaft of the flipper button fits through, with a recess on the outside for the bezel and a recess on the inside for the Pal nut.

I recommend a simpler approach: just drill **all the way through** with the **1 1/8" diameter** bit. This method is better in two cases:

- You're using the VirtuaPin bracket. That bracket provides its own collar on the inside, so you don't need the "neck".
- You're using the LightMite boards to illuminate the button. You don't want the narrower neck in this case, because it gets in the way of the LEDs. The wider opening lets you fit the LEDs into the opening alongside the button. And the LightMite board serves as the collar on the inside.

For either type of drill, fit the flipper buttons through the holes from the outside, and attach the Pal nut on the inside.

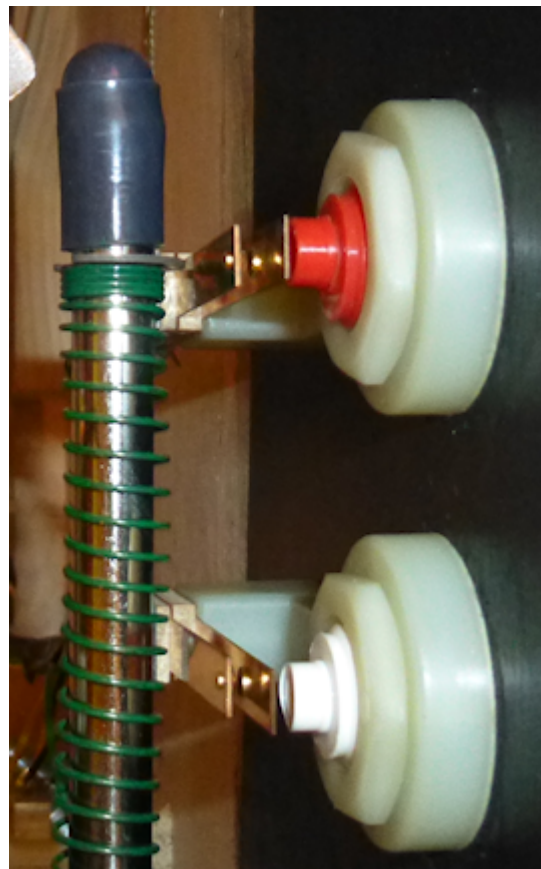
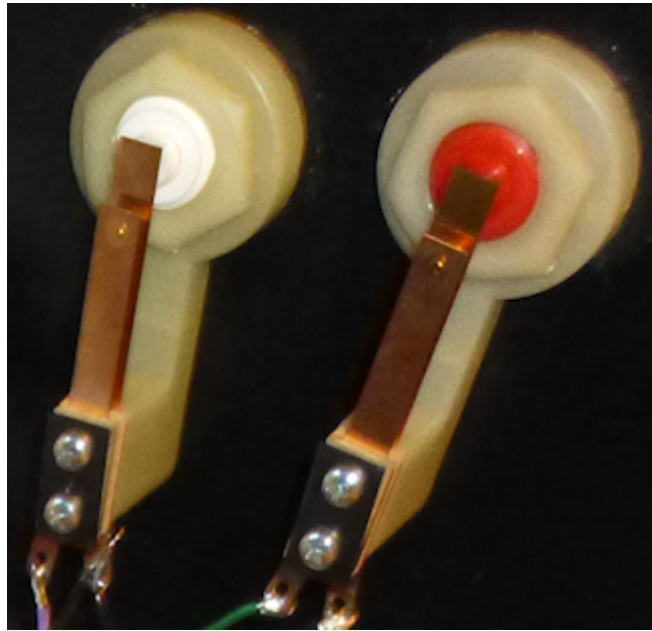


Leaf switch - VirtuaPin holder

The VirtuaPin switch brackets make it easy to install the leaf switches. They come pre-assembled with leaf switches properly aligned for the longer 1-3/8" buttons. To install them:

- Fit the flipper button into the cabinet hole
- On the inside, fit the switch bracket over the button shaft
- Slip the Pal nut onto the end of the button shaft and tighten

They look like this when assembled:



These **only** work with 1-3/8" flipper buttons (the "longer" length buttons).

These **won't** fit with the LightMite boards for illuminating the buttons. You'll have to use the wall mounting (below) for that.

Leaf switch - cabinet wall mount

If you're using the standard 1-1/8" flipper buttons, or if you want to use the LightMite board to illuminate the buttons, you can't use the VirtuaPin switch holders. Fortunately, it's fairly easy to mount the switches directly to the cabinet wall instead.

For the installation procedure, see "Flipper buttons" in Chapter 26, Inside the Cabinet.

Switch gap adjustment

Leaf switches can be finicky about the size of the gap between the contact points. If you see any flaky behavior from your flipper buttons, such as weird auto-repeats, or flippers flipping randomly while you're holding the button down, you might need to make some adjustments to the switches. See "Adjusting the leaf switch gap" in Chapter 26, Inside the Cabinet.

Single or double leaf switch?



Single-contact leaf switch (left) and double-contact switch (right)

For most people, the single-contact leaf switch is all you need.

The double-contact switches are used in real machines with extra flippers on one or both sides, such as the upper flipper on *Funhouse* (Williams, 1990) or the double right flippers on *Aladdin's Castle* (Bally, 1976). On those machines, they wired each flipper to a separate switch contact. This lets you control each flipper separately *with the single button*, by carefully modulating how far you press in the button.

I don't think most pinball players are aware that this is possible. So most people would never miss it on a virtual cab. But if you're one of those super-skillful players who knows how to take advantage of this feature - or maybe if you just aspire to get there someday - you'll undoubtedly want to re-create this feature on your cab. It'll certainly impress your pinball nerd friends if you include it.

As a practical matter for virtual cabs, the double-contact feature is *not* an inherent part of Visual Pinball or other simulators. Visual Pinball tables *can* take advantage of it, but they have to be specially programmed to do so on a table-by-table basis. This is an uncommon feature even among cab builders, so most table authors have no idea it's even possible, and it's not widely implemented in existing tables. If you're a big fan of the feature, and you know your way around Visual Pinball and its scripting language, you can add the feature to tables yourself with a little work.

In Visual Pinball tables that don't specially implement the double-contact feature, as

well as in most other simulators, extra flippers will usually just be tied to the main flipper button input, so you'll see the same effect as if you had a regular single-contact switch. In other words, both flippers on each side will fire when you press the button far enough for the first level of the double switch to make contact, and the second level won't do anything at all.

Flipper feedback effects

Most cab builders these days want some kind of feedback effect for the flippers, usually with contactors or solenoids, to reproduce the palpable "thunk" that the flippers produce on a real pinball machine. Some new cab builders imagine that it's necessary to add some extra wiring between the flipper buttons and the contactors or solenoids in order to achieve the feedback effect.

If you're planning to include Chapter 46, DOF in your setup, no extra wiring is necessary to achieve flipper feedback effects. DOF will automatically fire your flipper solenoids at the appropriate times to match the flipper action in the software simulators. This is the best way to handle flipper feedback effects, because it will make the flipper effects exactly match the on-screen effects in the simulation.

If you're not using DOF, though, you can add some rudimentary feedback effects by hard-wiring your flipper buttons to flipper solenoids or contactors. This requires some extra parts, because of the high voltages needed for the flipper solenoids. You can't just directly wire both the solenoids and the key encoder to your flipper switches, because the solenoid voltage will damage the key encoder. The extra wiring is described under Chapter 59, Flipper button feedback control in Chapter 59, Flippers, Bumpers, and Slingshots.

Wiring

For a single-contact leaf switch, wire one contact to the "common" or "ground" terminal your key encoder, and wire the other to a unique input for the button. The order of the terminals doesn't matter.

For a double-contact switch:

- Connect the "common" or "ground" key encoder input to the **middle** blade of the switch
- Connect the regular "Flipper Button" input to the blade **closest to the button**
- Connect the "Second Flipper Button" input to the blade **farthest from the button**

Typical key assignments

Left flipper: the left "Shift" key

Right flipper: the right "Shift" key

Left MagnaSave: the left "Control" (Ctrl) key

Right MagnaSave: the right "Control" (Ctrl) key

Second-level contact in a double leaf switch, left side: "L" key

Second-level contact in a double leaf switch, left side: "R" key

Optical interrupter switch

Okay, this is going to seem crazy after all of that lecturing and haranguing earlier about how *you must use leaf switches*, but...

The pinball manufacturers stopped using leaf switches in the early 1990s.

That's when they switched to "opto-interrupter" switches. An opto-interrupter is basically an "electric eye" type of switch, with a light beam and a receiver that detects if anything is in the way of the beam. In the case of the new flipper switches, this is all packed into a compact little unit about a centimeter wide, with a thin gap between the light source and receiver. There's also a little piece of plastic that sits in the gap, blocking the light from hitting the receiver. When you press the flipper button, it moves the little plastic shield out of the way and lets the light hit the receiver. That's what activates the switch, taking the place of the direct electrical contact between the two metal blades in the old leaf switches.

The first question is, why did they make this change? Reliability, mostly. The metal contact points on a conventional switch tend to get dirty over time from dust and grime in the air, plus the metal can oxidize and abrade. This all makes the contacts become less conductive as they age, which can weaken the flippers or make them operate sporadically. The leaf switch blades also tend to lose some of their elasticity and bend out of alignment from the constant back-and-forth flexing. Arcade operators always had to spend lots of time cleaning and re-aligning flipper leaf switches to keep their machines working properly. The optical switches, in contrast, are essentially maintenance-free. Arcade operators like that feature because it keeps their machines working and earning money more of the time, and reduces their repair budgets. Since arcade operators were their main customers, the pinball makers often designed around what the arcade owners wanted, and in this case, opto switches were desirable because of their improved reliability.

Second question: do the opto switches "feel right"? In other words, do they have the same force feedback properties and switching action as traditional leaf switches? In my opinion, yes. Some people disagree and believe the opto switches have a different and inferior feel compared with leaf switches, but I can't personally perceive a difference. I have real machines with both kinds of flipper switches, and they feel about the same to me. In fact, I owned both kinds of machines for many years before I even knew about the different switch types. If you look at the innards of the opto switches, you'll see that their construction is very much like that of leaf switches, physically: a flexible plastic blade takes the place of the metal leaf, and the motion of that blade controls the motion of the light shield in the optical portion of the switch. The mechanical action is essentially the same as for a leaf switch. The electronics are different, obviously, but I don't think that was ever much part of the feel; the mechanical action is what matters.

Third question: can I use these in my virtual cab? Yes, with some extra work getting the wiring right.

Fourth question: *should* I use these in my virtual cab? Probably not. They're more expensive and more complex to install, and I don't think there's enough of a benefit for a virtual cab. Remember that the main benefit of the optos is that they're low maintenance. That's a big deal for an arcade machine that's getting played all day every day. It's less of a concern for a home machine, though, since we don't put nearly as much mileage on it. The electronics in a virtual cab aren't as hard on the switch, either, since we're using low voltages. The leaf switches on the old pinball machines directly switched the 50V coil power, and that high power caused much more rapid oxidation of the switch contacts than you get with the 3V logic switching we use. What's more, you can get leaf switches with gold-plated contact points, and since gold doesn't oxidize, the contact points won't degrade from age alone.

Fifth question: how can I use these in my virtual cab? If you really want to use them despite the dubious benefits for home use, it's not too complicated. The part to buy is the WPC Fliptronic opto board and switch assembly, Williams/Bally A-17316. (The

components are also available separately.) The opto switches should be compatible with most keyboard encoder devices; wire them as normal, connecting the Ground (GND) pins on the opto board to the Ground or Common terminal of the encoder, and connecting the SW1 (switch 1) and SW2 (switch 2) pins of the opto board to the flipper button inputs on the encoder. These boards have two optos to provide the same effect as a double-contact leaf switch; if you only want a single-contact switch, connect SW1 and leave SW2 unconnected. You also have to supply the board with a +12V power supply connection to the labeled pin; this powers the light source for the optical switch.

Tilt bob

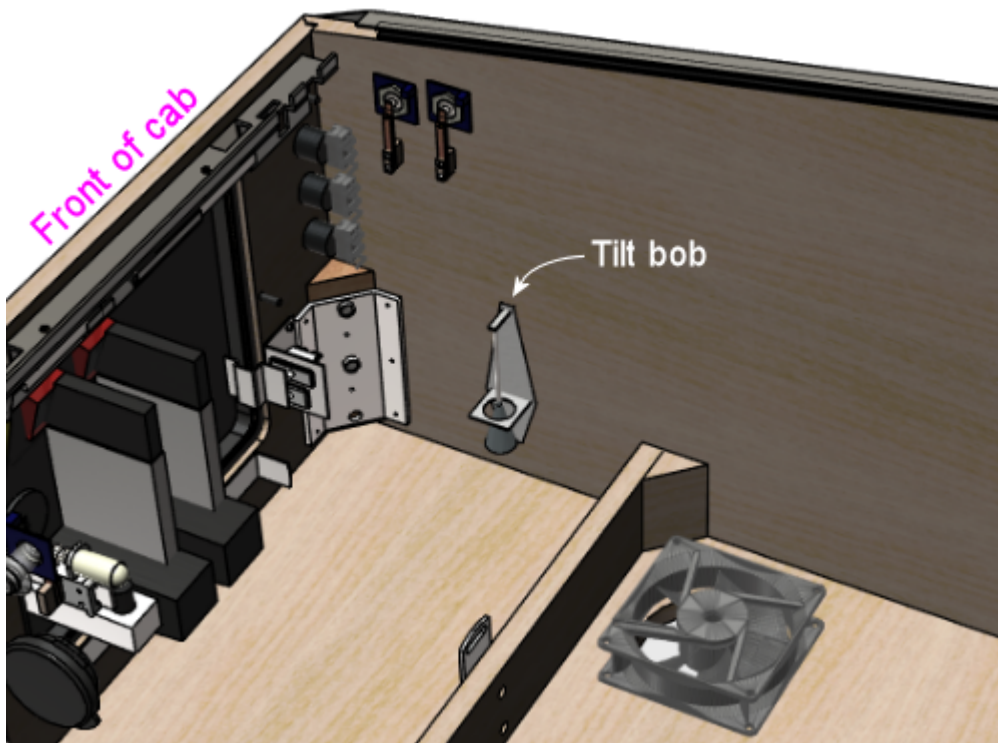
The easiest option is to buy a complete tilt bob assembly; look for part A-15361 at the pinball vendors. You can also buy the parts separately:

- Mounting plate A-15360
- Upper bracket 01-3444
- Lower bracket 01-3445
- Pendulum 03-8668
- Plumb bob wire 535-5319-02
- Plumb bob 535-5029-00

You'll also need two #6-32 x 3/8" machine screws to assemble the pieces to the mounting plate, and two #6 x 1" wood screws to mount it to the cab wall.

You can skip the mounting plate if desired and simply screw the two brackets directly into the cab wall. The mounting plate just makes it easier to get everything aligned properly.

In the real machines, the tilt bob is usually installed on the left side wall near the front of the cab.

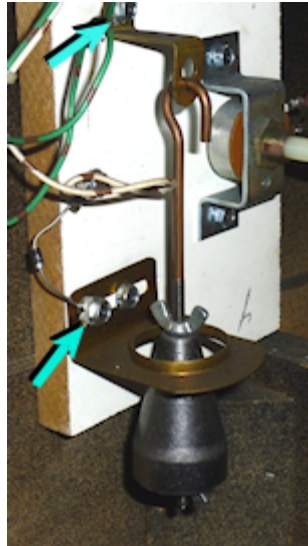


When installing, you should have the machine set up with its legs attached and leveled, so that the machine is sitting at the same angle that it'll have in regular use.

Angle the tilt bob so that the pendulum is centered in the ring when at rest.

Adjust the sensitivity of the tilt by moving the plumb bob up or down on the hanger wire. (Loosen the thumb screws to free the plumb bob to move up and down; tighten them again when the plumb bob is at the desired position.) A tilt is signaled when the plumb bob comes into contact with the surrounding ring, so it increases the tilt sensitivity when you move the bob upwards on the wire (reducing the margin between the bob and the ring). You can always come back and adjust the sensitivity after play-testing if it's too sensitive or too permissive.

To wire the tilt bob, connect one wire from the keyboard encoder to the terminal on the upper bracket, and connect the other wire to the terminal on the ring bracket.

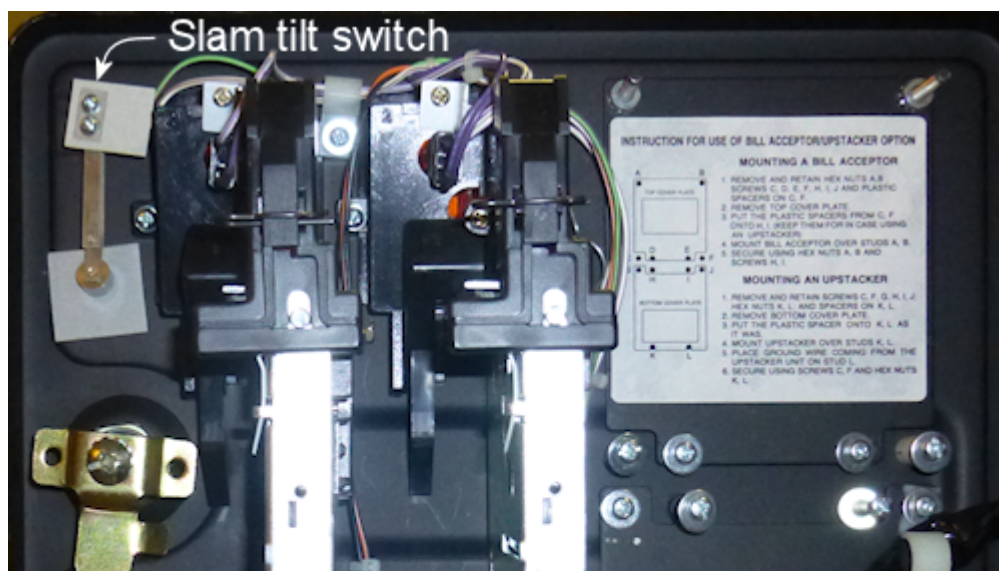


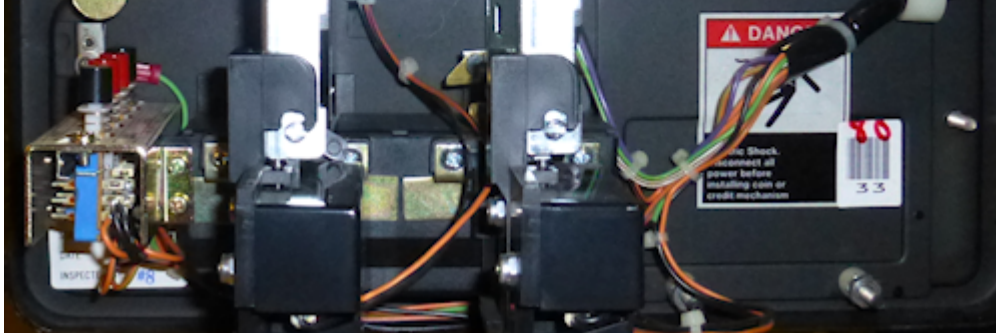
The standard keyboard assignment for the tilt bob in Visual Pinball is the letter "T" key. But some extra configuration is required! See "How to configure VP for a tilt bob" in Chapter 36, Nudge & Tilt.

Slam tilt

A slam tilt switch is a leaf switch with a big weight slug attached to the main leaf. This detects excessive jolts to the cabinet via the mechanical inertia of the slug.

WPC coin doors normally have a slam tilt switch pre-installed, so all you have to do is connect its wires to your keyboard controller. See Chapter 40, Coin Door for details on how to access the pre-installed wiring in the WPC door.

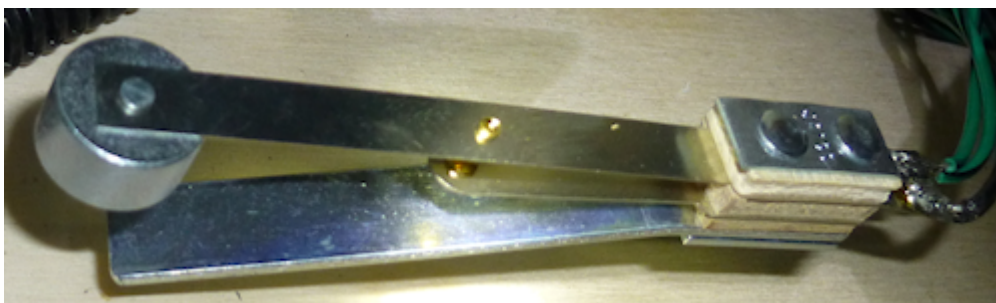




Slam tilt switch in a WPC coin door (interior side of coin door shown). The WPC doors typically have the switch pre-installed.

If your coin door doesn't already have a built-in slam tilt switch, and you want to install one separately, look for these parts at the pinball vendors:

- Weighted leaf switches: B-8372, 27-1066, 502-5032-00
- Mounting bracket (should be usable with any of the above): 01-1168
- Weighted switch with mounting bracket (pre-assembled): B-9141
- Planetary Pinball Supply weighted switch with mounting bracket: A-17195



Slam tilt switch with bracket, mounted to cabinet floor. It'll also work mounted to the front or side walls. Install near the front of the cab.

Mount the switch to any convenient spot on the cab floor, front wall, or one of the side walls, close to the front end. It should be within a foot or so of the front so that it'll be triggered if the front is lifted and dropped.

To wire, simply connect the wires from your keyboard encoder to the two leaf switch terminals.

The standard keyboard assignment for a slam tilt in VPinMAME is the "Home" key (the cursor navigation key).

Service buttons

If you have a WPC coin door with built-in service buttons, the buttons can be connected through the coin door's wiring harness. See "Wiring to the key encoder" in Chapter 40, Coin Door for details.

The Stern/SuzoHapp coin doors don't typically come with service buttons installed. You can separately buy a Stern 4-button service assembly (Stern part number 515-1963-00) that fits these doors. These come with four pushbuttons that you have to wire yourself. Connect a wire from the Common/Ground of your keyboard encoder to one terminal of each button (you can daisy-chain this connection across the buttons - there's no need to run a separate wire for each button back to the encoder). Connect the other terminal of each button to a unique key input on the

encoder.

The standard key assignments for these buttons are:

- Green button = Cancel = "7" (the "7" keyboard key)
- Left red button = -/Down = "8" (the "8" keyboard key)
- Right red button = +/Up = "9" (the "9" keyboard key)
- Black button = Enter = "0" (the "0" keyboard key)

If you're not using a standard coin door, but you want a service button panel somewhere, you might be able to adapt the Stern 4-button panel above to another mounting location. Or you could just get some small pushbuttons and attach them somewhere convenient with an improvised mounting. Wire them and assign keys the same way described above.

Coin door position switch

This is covered in detail under "Coin door position switch" in Chapter 40, Coin Door.

35. Button Wiring

In the previous section (Chapter 34, Cabinet Buttons), we looked at all the types of buttons that you might include in a virtual cab, and how to do the basic installation and wiring for each type. Now we'll look at the other end of the wiring: how you connect these buttons to your PC so that they can interact with the software.

If you're using a Pinscape Controller (either standalone or with the expansion boards), you might want to skip straight to Chapter 110, Pinscape Button Inputs, since that's a more streamlined How To guide for the Pinscape controller specifically. Likewise, if you're using one of the commercial encoders, read through its documentation for a quick start with its wiring setup.

The basic approach that we use to connect the buttons to the PC is to make them act like keyboard keys. To do this, we need a device known as a "key encoder", which is essentially the same circuitry that's inside an actual PC keyboard to convert physical key presses into USB signals that Windows can read.

In fact, in the early days of virtual cabinets, a lot of people *literally* used the circuitry from PC keyboards, disassembling an old keyboard and soldering their button wiring to the key switches inside. That was clever, but it was also rather difficult. If you've ever opened up any modern electronics, you know how microscopic and delicate everything inside is these days.

Fortunately, we no longer have to resort to such jury-rigged salvage jobs. There's a much easier way now: you can buy or build a dedicated key encoder built specifically for wiring arcade buttons. There are several turn-key commercial products available, or you can fairly easily build your own using the Pinscape Controller plans.

Joystick encoders vs keyboard encoders

Note that some encoders send button presses to the PC as joystick button commands instead of keyboard keys. These work exactly like keyboard encoders in terms of all of the physical setup and wiring, so everything in this chapter applies equally well to joystick encoders. For the most part, we use the term "keyboard encoder" to refer to both types, since they're really no different other than the USB commands they send.

(In fact, some controllers use both types of commands. The Zeb's Boards plunger controller uses a mix of keyboard and joystick keys for its button inputs. The Pinscape Controller lets you use either type for any button, letting you configure the selection for each button individually.)

Key encoder options

Several good commercial and DIY options are available. The commercial products are more expensive, but require little to no work to assemble or configure. The DIY options are cheaper but require some setup work.

- Standalone KL25Z: DIY. If you only need keyboard input (and not any feedback controller features), a bare KL25Z loaded with the Pinscape software is all you need. Fairly easy DIY setup and configuration, with no electronics assembly involved. The default configuration has 24 button inputs that can be mapped to any keys or joystick buttons, but you can assign more ports as button inputs (up to about 50 maximum) if you're not using them for other purposes.
- Pinscape expansion boards: DIY (very). The same Pinscape key controller

functions as with the standalone KL25Z version are also there if you're using the expansion boards. 24 button inputs by default, with four more unassigned pins available. Requires extensive electronics assembly work and fairly easy software setup.

- Zeb's Boards plunger kit: Commercial. Provides inputs for 20 buttons.
- VirtuaPin plunger kit: Commercial. Provides inputs for 16 buttons.
- i-Pac 2: Commercial. 32 button inputs.
- i-Pac 4: Commercial. 56 button inputs.
- i-Pac Ultimate I/O: Commercial. 48 button inputs.

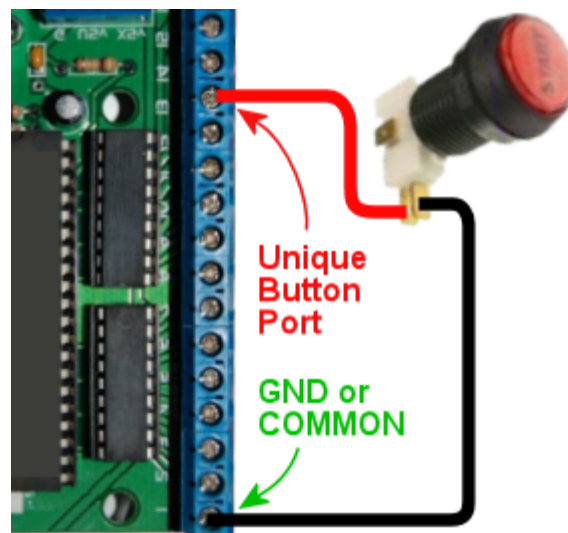
See Chapter 12, I/O Controllers for more details on the controller options.

Basic wiring plan

As with all wiring, make sure all power plugs are completely disconnected while you're working.

All of the key encoders (commercial and DIY) use the same basic wiring plan, so we'll show you how this works in general. You'll have to look at the instructions for your specific device to identify the actual physical terminals that we're talking about.

Here's the basic plan:



- At one end, you have the button. Electrically, each button is really just a switch that connects two wires together when pressed. So at the button end, we're connecting two wires, one to each terminal of the switch.

What about buttons with more than two terminals? No problem: you still only connect two wires. It's just a matter of figuring out which two terminals to connect. See "How to identify button terminals" below for help figuring out which two terminals to use.

(Any additional terminals on a button have other purposes that we don't care about when connecting the key encoder, so you can just ignore them for now. We might connect them to something else later, though. For example, an illuminated button with a lamp inside has terminals for powering the lamp, which we'd want to connect to the feedback device controller. But that's a separate matter for another chapter! For now, you can just leave any extra terminals disconnected.)

- At the other end, you have the keyboard encoder. We take those two wires that we connected to the button, and connect the other end of each wire to a terminal on the keyboard encoder.
- One wire to every button always connects to the "Common" terminal on the key encoder. Depending on which encoder you're using, this might be labeled GROUND, GND, COMMON, CMN, or C. Check your device's documentation if you're not sure which one that is.

Some key encoders have two or more Common terminals. If so, you can use any of them. They're all wired together internally so it doesn't matter which one you choose. Some of the controllers provide multiple Common terminals as a matter of convenience, to give you more flexibility in how you wire everything.

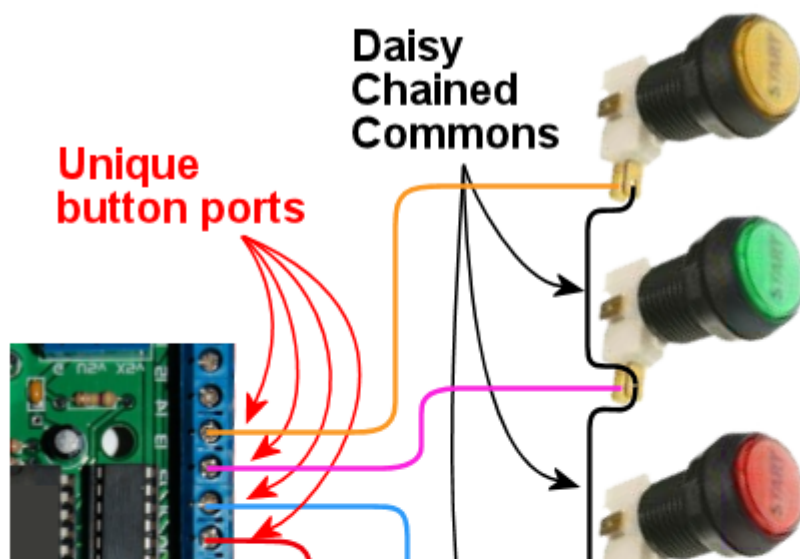
- The other wire to each button connects to a **unique button port** on the key encoder. On dedicated key encoders like the i-Pac, these are often labeled with pre-assigned arcade button functions, so you might see labels like "Start", "Player 1", "Coin 1", and so on. You should check your encoder's documentation to figure out which input is appropriate for which pin cab function. If you're using a Pinscape Controller, see Chapter 110, Pinscape Button Inputs.

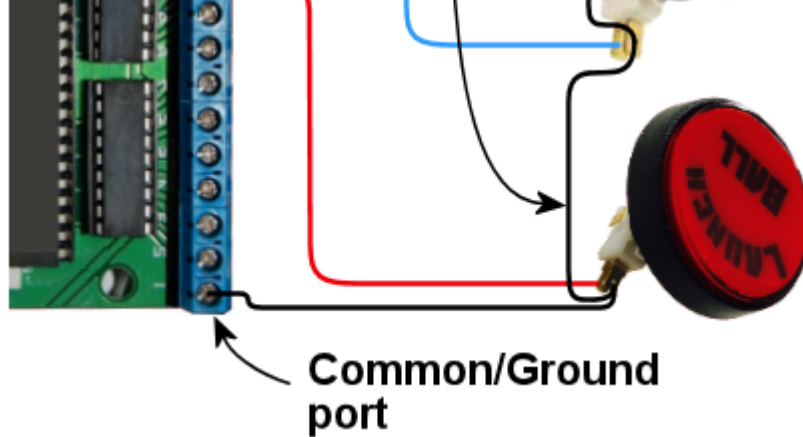
If your encoder lets you customize the keyboard mapping (the i-Pac and Pinscape allow this), it really doesn't matter which button you wire to which unique input terminal, because you can assign any key to any button later in the software setup. The function labels on the i-Pac are just the defaults.

If you're using a controller with a fixed set of key assignments (like the Zeb's Boards plunger kit), you'll need to wire each key to the port that matches the function. Your encoder's documentation should list the pre-defined keyboard assignments. See "Common key assignments" below for a list of the keyboard keys normally used in pinball software - that should help you match up each button to its appropriate port on your encoder.

- Every **unique button port** terminal can only connect to **one** button. That's why we call it "unique"!
- The COMMON or GROUND terminal will connect to **every** button in your system.

You don't have to connect a separate wire from every button all the way back to the Common/Ground terminal on the encoder, though! It's often easier to "daisy chain" the common, by connecting it from button to button:





All of the commons are connected together electrically, so it doesn't matter whether they connect back to the encoder with a separate wire or if they connect to each other in a daisy chain like this. You can mix and match the two approaches, too! Just connect each button's Common/Ground terminal to whatever's closest physically in the cab. If there's another button nearby, connect to the other button's Common/Ground. If the key encoder is closer than any button, run a wire back to the key encoder.

What type of wire should I use?

Any ordinary stranded hookup wire will work fine here. These are all low-voltage, very low-power connections, so you don't need anything beefy. 24 AWG is fine. (Thicker wire is more expensive; there's no reason to waste money on thicker wires here.)

I like to use a mix of insulation colors to make it easier to trace wires through the cab when necessary. I'd use one color for all of the Common/Ground wires - black, if you want to follow the normal conventions - and a mix of other colors (never the same as the Common/Ground color) for the unique button port wires.

See Chapter 71, Wire for more on wire selection.

Software setup

All of the key encoders emulate either a keyboard or a joystick, so there's no need for Windows device drivers. Windows should automatically recognize them as soon as you plug them in to the USB port or keyboard port.

If you're using a Pinscape Controller, you can configure how the buttons are mapped to keyboard keys or joystick buttons. Refer to Chapter 110, Pinscape Button Inputs.

Some of the commercial controllers (such as the i-Pac) also have their own setup programs that let you configure the keyboard mappings. Refer to your controller's documentation.

If your controller uses joystick buttons instead of keyboard keys, you'll have to configure Visual Pinball and any other pinball software you use to recognize the joystick buttons. For VP:

- Run VP without loading a table (just open the blank editor)
- On the menu, select Preferences > Keys in VP 9, or Preferences > Configure Keys, Nudge, and DOF in VP 10
- In the Button Assignments section, set the drop list for each button function to

match the joystick button number of the corresponding button, as you wired it to the key encoder. For example, if you wired the left flipper button to your key encoder port that sends Joystick Button 3 as its input, you'd set the drop list under Left Flipper to "Button 3".

Common key assignments

Here's a list of the key assignments that most of the PC pinball simulators uses by default. Most of the software gives you a way to change the key assignments, but it's always easier to use the defaults as much as possible. Fortunately, almost all of the PC pinball simulators have used the same core key assignments for decades, so you won't have fight with the software too much if you stick to the standard keys.

Function	Key	Notes
Start	1	
Exit	Esc	
Extra Ball	2	
Left Flipper	Left Shift	
Right Flipper	Right Shift	
Left MagnaSave	Left Ctrl	
Right MagnaSave	Right Ctrl	
Launch Ball	Enter	
Left 2nd flipper	L	For double-contact leaf switches, rarely used; see "Single or double leaf switch" in Chapter 34, Cabinet Buttons
Right 2nd flipper	R	For double-contact leaf switches rarely used; see "Single or double leaf switch" in Chapter 34, Cabinet Buttons
Coin In (main/left slot)	3	Used in VP/VPinMAME; "5" is more typical in other games
Coin In (middle coin slot)	4	VP/VPinMAME only
Coin In (right coin slot)	5	VP/VPinMAME only
Coin In (fourth slot/dollar bill)	6	VP/VPinMAME only
Tilt bob	T	See "How to configure VP for a tilt bob" in Chapter 36, Nudge & Tilt

Slam tilt	Home	
Open/close coin door	End	
Service Escape/Exit	7	
Service Down/-	8	
Service Up/+	9	
Service Enter/Select	0	
VP Volume Up	+	Only used in VP
VP Volume Down	-	Only used in VP

Note that the VP Volume Up/Down key assignments really aren't very useful. Those just adjust the relative volume of VP's table effects, and VP doesn't remember the settings across games, so you have to keep adjusting them over and over if you want to use them. It's pretty worthless. If you want general volume control buttons, I'd recommend skipping VP's keys and assigning the "Media Volume Up" and "Media Volume Down" keys instead, which Windows will use to adjust the master system volume level. That at least sticks across program sessions. Or better yet, use something like PinVol, which gives you finer controls that let you set per-game volume levels that are restored each time you return to a game.

How to identify button terminals

Electronically, every button is just a switch - basically a little gap between two pieces of wire that you can open and close. When the gap is open, the two wires are disconnected from each other, so no electricity can travel between them. When the gap is closed, the wires touch, conducting electricity through the switch.

So for each button, you need to start by identifying the two terminals for its switch.

This is trivial for some buttons, because two terminals is all they have! But a number of common arcade button types have multiple terminals, which makes it a little harder.

Let's take a look at some common button types and how to identify their switch terminals.

Arcade pushbuttons

This is the type you'll probably use for the front panel buttons on your machine. These combine a microswitch and a light bulb socket into a small plastic base. The complication is that they add two extra terminals for the lamp power wires, so you have to figure out which wires go to the lamp and which go to the switch.

Most of these have five terminals, arranged like this:





You should be able to find markings on the plastic body for at least two of the switch terminals, one labeled "C" or "COM", and the other labeled "NO" or "NC". Match the markings to the diagram above, and take into account any changes.

- Connect the Ground/Common wire from your key encoder to the "C" or "COM" terminal on the button
- Connect the other wire to the "NO" (Normally Open) terminal

If there aren't any markings, or the terminals are laid out differently, you might have to resort to the experimental method described below.

What about the lamp terminals? That's a whole separate subject, because even though the lamp is part of the button assembly, it's controlled by a whole separate electrical system on your virtual cab. That's covered in Chapter 55, Button Lamps.

Microswitches

These come in small black plastic cases with two or three terminals. There's usually a metal lever on the top that actuates the switch, but sometimes the switch paddle is just a little bump on top.

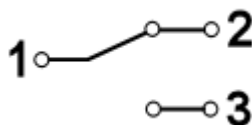


These usually have some kind of markings on the case near the terminal legs telling you how the terminals are wired inside the switch. The markings might only be raised or embossed in the plastic rather than inked, so they can be hard to see. Try looking at the switch under a strong light if no markings are apparent.

- **NO, NC, C (or COM).** Look for these markings next to the terminal legs. These stand for **N**ormally **O**pen, **N**ormally **C**losed, and **C**ommon. The C or COM label might not be there at all, but if the other two are marked NC and NO respectively, the unmarked leg must be the Common.

For this type of marking, connect the Ground/Common wire from your key encoder to the **C** or **COM** terminal, and connect the other wire to **NO**.

- **Numbered terminals.** Look for a small "1", "2", and "3" near each terminal leg. If you find these, you should also find a diagram printed (or embossed) on the case that looks something like this:



The "1", "2", and "3" lines represent the terminals. The diagonal line in the middle represents the moving contact in the switch. The diagram is telling you that normally, the moving contact connects between 1 and 2, but that when you press the switch, the moving part tilts the other way so that it connects 1 and 3. Connect the Ground/Common wire from your key encoder to terminal "1", and connect the unique button input on the encoder to terminal "3".

Note that the numbers themselves aren't what's important here: your switch might use different numbering or different labels entirely. The relationship between the terminals is what's important.

Coin door buttons: If you're using a real pinball coin door, it probably has several buttons and switches built in:

- Service panel buttons. A set of 3 or 4 pushbuttons inside the door. On a real machine, these access the operator menu.
- Slam tilt switch. This is a secondary tilt switch that's activated by hard jabs to the front of the machine.
- Coin chute switches. Each coin chute has a microswitch that's triggered when a coin is accepted.



You can connect all of these switches to your key encoder in the same way as other buttons. Visual Pinball has keyboard equivalents for all of these functions, so connecting these switches lets you access each function the same way you'd do it on a real machine.

Things get a little complicated at this point, though. The snag is that there are several types and several generations of coin doors available, and each one is wired differently. In particular, each type has its own special type of connector. You can find more details on the common types of connectors in Chapter 40, Coin Door.

If you can't find a wiring diagram for your specific coin door type, you can use the experimental method described below to trace the wires.

Once you identify the button and switch wires, connect them to the controller just like any other buttons. The coin door wiring usually includes a single common wire that connects to one terminal on each switch, plus one unique wire per switch. That's exactly how the basic key encoder wiring is set up, so just connect the coin door common wire to the key encoder's Ground/Common wire, and connect each unique switch wire to a separate button port on the encoder.

Coin door open switch: On a real pinball machine, there's a switch that detects when the coin door is open. Pinball ROMs use this to control access to the operator menus, so I'd recommend including one in your build if you're using a coin door. The Chapter 40, Coin Door chapter has suggestions for what kind of switch to use and how to mount it.

Once you have a switch set up, wire its "Common" terminal to the key encoder Ground/Common, and wire its **NC** or **Normally Closed** to a button port on the encoder. Note that this is backwards from most buttons, where you wire the Normally Open terminal. The reason for the reversal is that the geometry of the installation is kind of backwards: when the door is closed, it pushes down on the switch paddle, so the switch is "on". When the door is open, it releases the paddle, so the switch is "off". But we want Closed to read as "off" and Open to read as "on". The easy way to accomplish this reversal is to use the Normally Closed half of the switch, which reports the opposite status of the Normally Open side.

The coin door open button needs a little bit of special treatment in the software setup. On a real pinball, the coin door switch is just a switch: it's ON when the door is open and OFF when the door is closed. But Visual Pinball, by default, treats it as a toggle button, not a switch: push the button to open the door, push the button again to close the door. You can change this handling in VP with a little scripting work - see "Setting up the coin door switch in VP" in Chapter 40, Coin Door.

Tilt bob

You can connect a tilt bob to a button input, to detect TILT conditions the same way a real machine does.

Note that the tilt bob **isn't** there for "nudging" in the simulation. It's far too blunt an instrument for that. The tilt bob is there to serve the same purpose it does in a real machine, which is detect the overly aggressive nudging that counts as cheating. For the kind of nudging where you want to influence the ball motion in the simulation, use an accelerometer. That can differentiate between gentle nudges and hard nudges and everything in between. See Chapter 36, Nudge & Tilt for more.

To connect a tilt bob, simply connect the two usual wires from the key encoder (Ground/Common and a unique button input) to the two ends of the bob: one to the hanger hook, the other to the ring at the bottom. Standard pinball tilt bob assemblies have screw terminals where you can attach the wires - the arrows below show where.



Where to attach the switch wires to a tilt bob. Note that this is the tilt bob in a real pinball machine. If you're wondering about the diode visible in the photo, that's only there because of the "matrix" switching used on the real machines. It's not needed in a virtual cab.

Switches with four or six terminals

A switch with more than three terminals is probably a "double pole" switch, meaning that it has two separate switches inside, mechanically linked so that they turn on and off together. These can come with four terminals or six terminals.

If the terminals are marked, they should indicate some kind of grouping to let you know which terminals belong to which switch. For example, you might see something like "1NC - 1NO - 1COM / 2NC - 2NO - 2COM". All of the "1" terminals are part of the first switch, and the "2" terminals are part of the second. Alternatively, you might see a little switch diagram like the "numbered terminals" diagram above, but with

two copies of that circuit. The two circuits represent the two switches.

This type of switch connects to your keyboard controller with two wires (Ground/Common and a unique button input), just like any simpler switch or button. The trick is just to ignore that second switch, acting like the extra terminals don't even exist. As before, you just need to identify one pair of Normally Open contacts. You can leave the other terminals unconnected.

Identifying button terminals experimentally

If you have an unusual button that doesn't fit any of the styles above, and you can't find any markings, you can always use a voltmeter to identify the terminals.

In fact, even if you've identified a button's terminals based on markings, it's not a bad idea to double-check your findings with a voltmeter to make sure you read correctly.

If your voltmeter has a Continuity Tester setting, select that. On this setting, the meter should emit a beep when it detects a good connection. If you don't have a Continuity setting, use the Ohms setting instead, and read it like this: INFINITY (∞) Ohms means no connection, 0 Ohms (or close to 0 Ohms) means there's a good connection.

If you already think you know which pair of terminals to use, touch the meter's leads to the two terminals. It should read as infinity Ohms (no beeping if you're in continuity tester mode). Keeping the leads on the terminals, press the button. The meter should change to 0 Ohms or should beep in continuity mode.

If you have no idea which terminals are which, simply try each pair in turn until you identify the pair that behaves as just described.

36. Nudge & Tilt

Nudging and tilting are essential aspects of real pinball for any serious player. A simulation has to do these well to be convincing.

It's easy to set up good nudge and tilt handling on a virtual cabinet, thanks to a type of sensor known as an accelerometer. We'll explain the specifics of setting everything up shortly. First, though, some background information to explain why you want this.

Nudging vs tilting

Nudging and *tilting* are two separate things to pinball people, but this can be a little confusing because casual players often use them interchangeably. To be sure we're all talking about the same thing, let's define our terms.

Nudging is an attempt to influence the motion of the ball by pushing, bumping, wiggling, shaking, lifting, and otherwise moving the pinball machine. A reasonable amount of nudging is fair play. It's one more way of interacting with the game. It's integral to pinball's uniquely mechanical nature; it's part of what makes pinball different from video games.

Tilting is the result of too much nudging. A tilt occurs when the machine decides that the player is nudging too aggressively. The machine penalizes the player by immediately ending the current ball and canceling any bonus.

On real machines

It's hardly worth explaining how nudging works on a real machine, since it's intuitive everyday physics. However, if your past experience with pinball is entirely virtual, your intuition about it might be a bit distorted, since the PC pinball simulators tend to exaggerate it quite a bit compared to reality. You really should find a physical pinball machine and play around with it to calibrate yourself. The real thing is rather subdued and subtle compared to PC simulations, mostly because real balls and cabinets are a lot heavier than they act in most PC pinball games.



Tilting is a more interesting topic, since most people haven't had a chance to look inside a real machine to see exactly how it works. A real machine enforces the tilt limits with something called a "tilt bob". This is a simple mechanical sensor that goes back to the early days of pinball.

A tilt bob is a free-swinging pendulum surrounded by a metal ring. Both the pendulum and the ring are connected to opposite ends of an electric circuit, so if the two come into contact, it closes the circuit, just like tripping a switch. Shoving or lifting the machine makes the pendulum swing, and if it swings enough to come into contact with the ring, it triggers the tilt switch. The sensitivity of the tilt can be changed by adjusting the spacing between the pendulum and ring. The weight at

the end of the pendulum is usually cone-shaped, so the distance can be adjusted by moving the weight up or down on the pendulum arm. This lets operators make tilting easier or harder according to how much abuse they want their machines to put up with.

Newer electronic games usually give you warnings for the first couple of contacts between pendulum and ring, and tilt if you exceed the warning limit. The warning count is adjustable in the operator menu on newer games, giving the operator another way to adjust the tilt sensitivity. Older EM machines usually tilted immediately on the first contact.

The tilt bob is the standard on modern machines, but if you look at older machines you'll sometimes find other types of tilt switches, in addition to or instead of a tilt bob. For example, some 1960s machines had a ball that could roll up and down a track inside the cabinet, to detect if someone tried to lift up the front of the machine.

Slam tilt

In addition to the tilt bob, most real machines also have something called a "slam tilt" switch. This is a weighted leaf switch inside the coin door, which can be set off by sudden forward accelerations, such as someone kicking the front of the machine. It's really more of a "slam" detector than a "tilt" detector; you wouldn't set one off with any amount of nudging. The real machines include them mostly to deter excessive abuse by frustrated players.

I personally wouldn't go out of my way to include a slam tilt switch on a virtual cabinet, but if you install a real coin door, it'll probably have one built-in. Visual Pinball and PinMAME have support for this switch as an input, so you can attach it if you wish. See Chapter 40, Coin Door for more on that.

Virtual nudging, part 1: the sensor

In my opinion, there's only one type of nudge sensor worth considering: a digital accelerometer. If you're planning to buy one of the commercial plunger kits, it'll have one built in. Otherwise, I recommend buying a Freescale KL25Z (which costs about \$15) and installing the Pinscape software (free). The KL25Z comes fully assembled; all you have to do is plug in a USB cable. It has an excellent on-board accelerometer, and the Pinscape software is compatible with VP's nudge input system.



In the early days of virtual pinball cabs, digital accelerometers weren't readily available, so people came up with various other schemes. I wouldn't recommend any of these for a new build, but for the sake of historical interest, I've included a summary of older methods at the end of the chapter.

Virtual nudging, part 2: the software

Visual Pinball has two completely different ways of handling nudge input, to accommodate the two main ways VP is used: desktops and cabinets.

For desktop users, VP accepts "digital" nudges via the keyboard. VP lets you enter three types of keyboard nudges: forward (by pressing the space bar), left (by pressing /), and right (by pressing Z). This keyboard convention was invented by the earliest desktop PC pinball games, long before VP's time, and has been the universal standard ever since. When you press one of these keys, VP simulates a single sharp nudge in the corresponding direction.

I call these keyboard nudges "digital" because you can't control the intensity or duration. VP at least gives you a small amount of control over the direction, but even that's severely limited, in that there are just the three fixed directions to choose

from.

For cabinet users, VP accepts "analog" nudges, via the joystick USB interface. VP doesn't expect you to hook up an actual joystick; it just uses the joystick interface because it's a simple, standard device type on Windows. The way you use this is to connect a device that *pretends* to be a joystick, but actually sends accelerometer readings.

VP interprets joystick nudge input by reading the X and Y axes from the joystick, and treating them as the amount of X and Y acceleration. Importantly, this allows for varying degrees of strength in the nudging. It also allows VP to detect the direction of the motion - side-to-side and front-to-back. Since each axis is analog, these can combine to represent nudging in any direction and at any strength.

Virtual tilting

As with nudging, VP has two different approaches to tilting, geared respectively to desktop and cabinet users.

VP's default approach is geared towards desktop users, who use only simulated nudging via the keyboard. In this case, nothing "real" is moving, so VP has to judge when enough is enough entirely within the simulation. This is usually handled in the scripting (programming) code for each individual table, and it's usually based on a timing model where you're limited to a certain number of nudges within a certain time frame. If you press the nudge keys faster than allowed by this limit, the table tilts.

The scripted approach doesn't work for accelerometers, because the accelerometer input comes in through the joystick, not the keyboard. The accelerometer doesn't press the keys no matter how hard you shake the machine. To make up for this, VP has its own internal simulation of a tilt bob. Whenever the physical accelerometer detects motion, VP feeds the motion into its simulated tilt bob. When the simulated bob swings too wildly, VP sends a "center tilt" key press to the game script so that it can register the tilt.

There's a third approach that I prefer to either of these, which is to install a real tilt bob in your cabinet. This is the most realistic solution because it detects the physical motion of the cabinet exactly like the real machines do. This captures all of the nuances of your cabinet's actual motion, which is too complex for VP's simple model to simulate realistically.

Installing an accelerometer

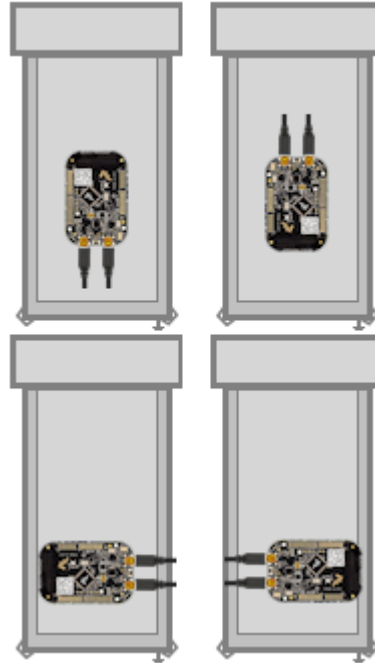
The Pinscape controller software and the commercial plunger kits are programmed to feed their accelerometer readings to VP using exactly this joystick interface. The whole setup is almost entirely plug-and-play with any of these devices. Just plug the device into your PC with a USB cable. Windows and VP should recognize it as a joystick.

The commercial plunger kits come ready to use out of the box. For the Pinscape controller, you'll have to install the Pinscape firmware, described in Chapter 89, KL25Z Software Setup.

For best results, you'll want to attach your device to the floor of your cabinet. The device should be fixed firmly in place, because you want it to move with the cabinet and experience the same accelerations that the cabinet does when the player nudges the machine.

If you're using a commercial plunger kit, check its documentation for the recommended positioning and orientation.

For the KL25Z, you should mount it flat on the cabinet floor, preferably near the middle front, with the edges of the KL25Z card parallel to the sides of the cabinet. As long as it's parallel to the sides, the orientation is up to you. You can position it with the USB cable connectors at the front, back, left, or right, whichever is most convenient. You just have to tell the software which way it's positioned so that it can adjust the readings accordingly.



Valid orientations for the KL25Z, as viewed from above. Position with the edges of the card parallel to the sides of the cabinet, around the front middle of the cab.

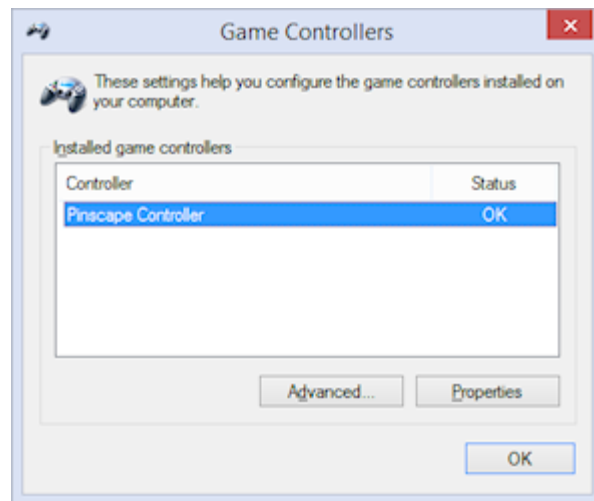
Do I need to add rubber foam to isolate it from vibration? Probably not. I've seen a lot of posts on the forums suggesting that you need some kind of soft padding to isolate the accelerometer from vibrations coming from the speakers, solenoids, and so on, as though it were an LP player in an audiophile's stereo system. But I think that advice is based on the mistaken idea that the accelerometer is "triggered" by motion. That might have been true of the older, kludgier nudge devices that people used before accelerometers became popular, but it's not true with accelerometers. An accelerometer is a measuring instrument, not an alarm. It isn't *triggered* by motion - it *measures* motion, quantitatively. It distinguishes between a little motion and a lot of motion, and all points in between, and the data it sends to the pinball simulator reflects the amount of motion measured, on a linear scale. It's not an all-of-nothing, on-or-off sort of thing. If the accelerometer picks up the tiny sub-millimeter motions of the cabinet from the speakers, it sends correspondingly tiny signals to the pinball game, and the pinball game responds with proportionally tiny effects on the ball's motion, which (when things are adjusted properly) are so small that they're invisible. There shouldn't be any need to filter them out. If you think about it, the steel balls in a physical pinball machine are themselves affected by all of the same sorts of cabinet vibrations - they're not wrapped in foam. If you install an accelerometer and find that speaker vibrations are in fact causing anomalous nudge input, my first impulse would be to reduce the "gain" setting in the pinball simulator. Small vibrations causing big effects suggests more than anything that the pinball simulator is amplifying the input way too much.

How to configure Visual Pinball for an accelerometer

Visual Pinball handles accelerometer input via the joystick interface. All of the pinball nudge devices (including Pinscape and all of the commercial plunger kits) are set up to work this way by default, so there's usually nothing you have to do with the device other than plug it in to a USB port. The only configuration you have to do is to VP itself.

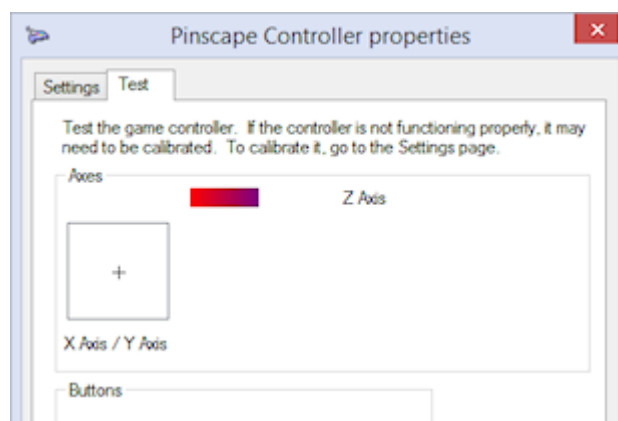
Quick device test: Before firing up VP, it's a good idea to make sure your accelerometer device is working properly. VP doesn't give you any feedback at all about whether there's even a device present, let alone if it's working, so you can save yourself some frustration by checking to make sure Windows recognizes the device and can see the acceleration input.

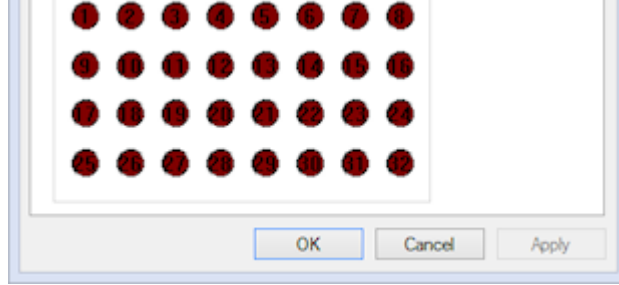
First, make sure the device is plugged in to a USB port. Now press Windows+R to bring up the Windows "run program" box, type in **joy.cpl**, and press Enter. This should launch the USB joystick control panel, which should show a list of attached joystick devices. Find your accelerometer device in the list. For example, if you're using Pinscape, you should see a "Pinscape Controller" entry in the list.



If you don't see your device listed, Windows didn't recognize it as a joystick. Check your device's documentation or contact the vendor for advice, or ask on one of the forums (e.g., the vpforums cab builders group). I'd advise against messing around with Device Manager or trying to install or update device drivers. Joysticks don't usually need device drivers in the first place, so that's almost never the problem (unless your device's documentation specifically says otherwise).

Double-click the list entry for your device. This will bring up the "Test" window, which lets you see the raw joystick input the device is sending to Windows.





Warning! DON'T use the "calibration" feature. Windows calibration isn't suitable for nudge devices; it'll distort the readings and cause erratic behavior in VP. See below for more information.

For nudging, the thing to pay attention to is the "X Axis / Y Axis" box with the little "+" inside. The "+" shows the current X/Y axis reading from the joystick, which is where the nudge device reports the accelerometer data.

If you haven't yet installed your device in the cabinet, you can pick up the device and tilt it in different directions. Gravity is a type of acceleration, so as you tilt the device, it should report an acceleration in whichever direction is pointing up. (That sounds backwards, I know. But you can thank Einstein for this bit of disillusionment in the name of science. It turns out that the right way to think about it isn't that gravity is pulling us down, but that the ground is pushing us up. That contradicts our subjective experience of it, I know, but only in the same way that the Earth going around the sun contradicts our subjective experience of that everyday phenomenon.)

If you've already bolted down the device inside your cab, you can test it simply by nudging the cabinet. You should see the "+" dance around when you push the cabinet, and the distance it moves from the center should be proportional to the strength of the push.

Don't worry too much about the particulars of the motion. The important thing is that you can make the "+" move left, right, up, and down in response to tilting the device. Note that many accelerometers are sensitive enough to pick up rather minute vibrations, so you'll probably see the "+" jiggling around a little bit even when you're not nudging the cabinet or tilting the device. As long as it's staying very close to the center, a little random motion is normal. However, the difference between the random motion at rest and the response to a nudge should be large and obvious: you shouldn't be seeing a lot of motion when the device is at rest, just a little random jiggling.

If the "+" is moving around as expected, the accelerometer working, and you can move on to setting up Visual Pinball. If you're not seeing any motion on the X/Y axis display, or the motion doesn't correlate with physical accelerations you apply to the device, something's not working properly. You might need to contact the vendor or ask on one of the forums for help.

Warning: DON'T use Windows calibration!

The Windows joystick setup dialog has a "Calibrate" button, and if you're like most people, clicking it will be all but irresistible. But resist! The Windows calibration is designed for actual joysticks. It's all wrong for nudge devices. If you run the calibration on a nudge device, it will screw up the nudge readings, and you'll see erratic behavior in VP. ¹

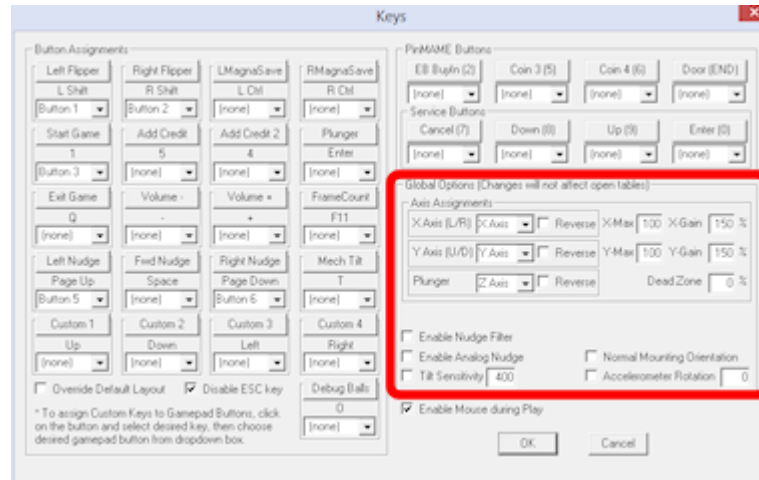
But don't worry, there's no permanent harm if



Ren tries to impress upon Stimpy the importance of not pressing the joystick

you did click the button at some point. You can easily undo it. Go to the Settings tab in the joystick dialog (the one pictured above), then click "Reset to default". That'll erase the Windows calibration data.

Setting up Visual Pinball: Start by bringing up the VP editor. (In some versions, you have to cancel out of the initial file selector dialog to reach the editor.) In the Preferences menu, select Keys to bring up the keyboard dialog. The accelerometer settings are in the "Global Options" area.



Here are the key things to set:

- **Enable Analog Nudge:** Check this box to enable accelerometer input. VP ignores the joystick input if this box isn't checked.
- **X-Gain** and **Y-Gain:** These determine the strength of the nudge effect in the simulation for a given physical acceleration. Higher numbers make the effect of the same nudge stronger, lower numbers make it weaker. In most cases, the X-Gain and Y-Gain numbers should match. Finding the ideal settings for your system requires experimentation. For now, start with the defaults and finish setting up the rest of the options. We'll explain how to find ideal settings below.
- **Enable Nudge Filter:** Check this box to tell VP to "filter" the raw accelerometer data to make the nudge effect more consistent and stable. The filter tries to cancel out certain types of systematic measurement errors that are common with these devices. Without the filter, the raw accelerometer data can make the ball "drift" as though the playfield were slightly tilted. The filter is optional, though, since any type of processing like this can introduce artifacts of its own. I recommend enabling the filter initially; you can always experiment with it later once you have the basic setup working to see if you prefer the unfiltered input.
- **Tilt sensitivity:** If you check the box, it enables a *simulated* tilt bob within VP, with the number specifying how easily it triggers a tilt. Higher numbers make tilts easier to trigger. **Disable this if you're using a real tilt bob**, since the simulated tilt bob is redundant.
- **Axis assignments:** In the Axis Assignments box, there's a drop list next to each of "X Axis (L/R)" and "Y Axis (U/D)" that lets you assign a different joystick control to the axis. In most cases, you should leave these at the default settings, unless your nudge device's documentation tells you otherwise. The defaults are the obvious mappings: "X Axis (L/R)" = "X Axis" and "Y Axis (U/D)" = "Y Axis". Make sure the Reverse boxes are un-checked.

After you set all of the options, click OK to close the dialog, then **close all VP windows** to exit VP. It's important to **completely** close VP after changing these settings, since VP won't reload the new settings until you close the program and launch a brand new session.

VP 9 vs VP 10: Before going on, one really important thing to be aware of is that VP 9 and VP 10 require radically different Gain settings. If you're using both versions on your system, you'll have to find the right gain settings for each version separately, because they don't translate across the versions. The rule of thumb is that VP 9 settings should be approximately 10 times higher than VP 10 settings.

For example, if a Gain of 100 works well for you in VP 10, you'll probably set the VP 9 Gain to about 1000. The ratio might not end up being exactly 10 to 1 on your machine, since the ideal settings vary by system and according to your taste, but it should be in that ballpark.

Finding the ideal X/Y Gain settings: I'm afraid I can't just give you one-size-fits-all numbers to plug in to the X/Y Gain boxes. The ideal settings for your system depend on the particulars of your machine: which nudge device you're using, your CPU speed, your graphics card, your personal taste, and even which games you're playing.

The way to find the right settings is by experimenting, by running a game and testing different nudges to see what kind of effect they have on the ball. It's easiest to do this when a ball is sitting in the plunger chute or captured with a flipper. Give the cabinet a push and see how strong the reaction is. Try different strengths of pushes and see if the reaction seems natural or not.

Follow this procedure to adjust settings:

- Open a game of your choice in the VP editor
- Run the game
- Test some nudges and see if the effect feels natural
- If the effect feels too strong (the simulation overreacts to slight nudges), you'll need to **reduce** the gain settings
- If the effect feels too weak, you'll need to **increase** the gain settings
- Quit out of the game and return to the editor
- Bring up the Keys dialog via the Preferences menu
- Increase or decrease the Gain settings as you decided above: try large changes at first (double or halve the settings, perhaps), and make smaller changes as you zero in on the sweet spot
- Click OK to close the dialog
- **Close all VP windows.** This step is crucial because VP won't load the new settings until you **completely** exit the program and restart it.
- Re-launch VP from the desktop and start over

While you're testing the nudge strength, also observe the *direction* of the ball's response and make sure it seems appropriate. A forward push should make the ball move up/down, not side-to-side, and a sideways push should make the ball move side-to-side rather than up/down. If these seem backwards, you might either have your device oriented incorrectly, or you might need to adjust the "Axis Assignments" in the VP setup.



Remove other joysticks: Visual Pinball won't work correctly for nudging if

you have multiple joystick devices connected to your system. Some game controllers that don't physically look like joysticks *act* like joysticks as far as VP is concerned, so if you're having any problems getting nudge working, try disconnecting all USB game controllers apart from your nudge device.

What's realistic?

The ideal strength of the effect is of course up to you. One of the great things about virtual pinball is that it doesn't have to be perfectly realistic: parts don't have to break, the playfield paint never has to get worn down, and nudge reactions can be as wild as you want.

If you want to calibrate for realism, though, it's a really good idea to find a real machine and play a few games, paying special attention to the way a real ball reacts to nudges. Run the same kinds of tests suggested above, such as nudging with a ball sitting in the plunger chute or trapped on a flipper. Gauge how much force it takes to make the ball jump a noticeable distance off the flipper when trapped. If you're accustomed to desktop pinball, you'll probably be surprised at how much force is required to get even a slight reaction on a real machine, let alone sending the ball flying an inch or two off the flipper the way a keyboard nudge does in desktop play.

One thing that can be hard to get accustomed to if you're coming from a desktop pinball background is the idea that accelerometers are analog devices. In desktop pinball, nudging is a "digital" action: you push a button and the ball jumps a certain fixed amount. In a virtual cab with an accelerometer, though, a nudge doesn't have a single fixed amount of force in the game. The simulated response should be proportional to the physical force you apply. Don't think of the cabinet as a giant space bar that you press to get that digital nudge. You shouldn't expect or want the ball to make that same fixed digital jump every time you give the cabinet the slightest touch. The reaction should be proportional to how hard you nudged. It's okay if there's no obvious reaction to a very slight nudge; go back to the real machine and see how much force it takes before you see any reaction at all.

It's fine to calibrate for an exaggerated version of reality if that's what you prefer, but it's still worthwhile to get a visceral idea of what the real thing looks like, as a reference point. I personally find that a slightly exaggerated degree of reaction feels about right on VP; calibrating for reality leaves things a little too flat in the virtual version.

What about interference from the shaker or subwoofer?

One of the frequently asked questions by new cab builders is whether cabinet vibrations from the game itself, such as from the audio system or from the shaker motor, will cause unwanted accelerometer feedback. This seems like a reasonable worry when you consider that accelerometers are designed to pick up tiny motions.

Here's a sanity-check question to ask yourself. Do real pinball machines have the same sources of vibration? Obviously they do. Do these same vibrations on a real pinball affect the ball noticeably? Obviously they don't. So, should vibrations that don't affect the ball in a real game affect the ball in a virtual game? Or put another way: is the simulation accurate if it responds differently from a real game to the same vibrations?

This brings us back to the point above in "What's realistic?", that virtual pinball tends to exaggerate the effect of nudging. If you *do* see interference from your shaker motor or audio system, it's a very good sign that you have the Gain settings turned up well above realistic levels. Go back to the accelerometer settings in VP and make further adjustments to find a happy medium for the Gain level: high enough that the

ball responds to your intentional nudges, but low enough that the ball doesn't go veering off every time the music plays loud.

In practice, some cab builders do have problems finding this happy medium in Gain settings. In my opinion, the accuracy of the accelerometer is the crucial factor here. I've found the KL25Z accelerometer to be excellent for virtual cab use, so if you're using a different nudge device and simply can't find the happy medium, you might consider adding a KL25Z with the Pinscape software. The KL25Z is inexpensive (about \$15), and the nudge feature is easy to set up, just a matter of plugging in the USB cable and installing the Pinscape software. The Pinscape software happily coexists with other I/O devices (LedWiz, PacLed, other plunger kits, button encoders, etc), so you can use it for its nudge features alone even if you've already decided on other devices for other functions.

Setting up FX2/FX3 to work with an accelerometer

Pinball FX2 and FX3 can also simulate nudging using an accelerometer, but they don't use the joystick interface that the Pinscape Controller and most other nudge devices use. Instead, they require input through the XBox controller interface.

To bridge the gap, there's a program called **x360ce** that can make a joystick device emulate an XBox controller. That can reportedly be used to make a joystick-based accelerometer work in FX2/FX3.

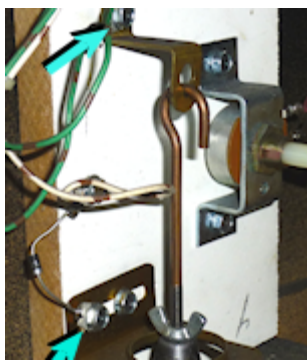
I don't use this in my own system, so I don't have any details about how to set it up. If anyone wants to write up instructions, I'll be happy to include them here.

Installing a real tilt bob

The best option for a tilt bob is to buy a real one from a pinball parts supplier (see Chapter 4, Resources). They cost about \$10 to \$20. You could also fashion one yourself, but the real ones are cheap enough that it's probably not worth the added effort.

On a real machine, the tilt bob is usually installed on the left side wall near the front of the machine. This is the best place for it because the player will primarily be nudging the machine near the front. If you haven't already installed your playfield TV and flipper buttons, be careful to pick a spot that won't get in the way of anything.

Electrically, wire the tilt bob just like a button or switch. As with all buttons, you'll run two wires between the bob and your key encoder device: the "common" or "ground" wire, and a wire connected to the input port you'll use for the tilt key. On the bob, one wire connects to top of the pendulum, and the other connects to the ring. It doesn't matter which wire goes to which end. The bob parts are usually all metallic, so you can attach the wires anywhere that's convenient. The standard tilt bob assemblies for real machines include screw terminals for the wires.





Screw terminals (arrows) for connecting switch wires to a standard pinball tilt bob assembly

If your key encoder has a pre-programmed port for "tilt" or "T" key, use that. If your encoder is programmable, attach it to any port, and program that port to send the "T" key.

Button wiring is described in more detail in Chapter 110, Pinscape Button Inputs.

How to configure VP for a mechanical tilt bob

As far as Visual Pinball is concerned, the tilt bob is simply another keyboard input. If you installed a physical tilt bob as described above, it will send a "T" key press to the PC whenever the tilt bob makes contact, as though you typed "T" on the keyboard.

VP didn't originally support real tilt bobs at all, since VP was initially designed for desktop PCs, before anyone even thought of virtual pin cabs. Support for tilt bobs had to be retrofitted into VP later in its evolution, and like most retrofits, the support isn't quite seamless. But with a little tweaking, we can fix that and make it work right for virtual cabs.

The way that VP handles the "T" key input is simply to pass it through to the Visual Basic script that controls the table. That's really what's "not quite seamless" about VP's handling of tilt bobs: most table scripts are programmed for desktop play, so they don't expect a real tilt bob to be present.

Fortunately, most table scripts use a single, shared script file for the nudge key handling. That means we can modify most of our installed tables simply by updating this one shared script file. What's more, we don't even have to edit the main shared script file by hand. The shared script has a "plug-in" design that lets us change some of its behavior by placing a script file with a certain name in a certain VP folder.

Just for reference, here's what the standard shared script does. When the "T" key is pressed, the default script performs a "fake" keyboard nudge, applying a brief acceleration to the physics model so that the ball's motion is deflected a little bit. The script also keeps a count of "T" key presses, and uses a timer to keep track of when they occur. If several "T" key presses occur within a few seconds, the script sends a "tilt switch" signal to the game's ROM. The counter and timer serve as a crude approximation of a real tilt bob: the idea is that too many of these fake nudges too quickly should count as a tilt.

Here's what we *want* to happen instead. Since we're using a real accelerometer, we don't need or want the fake nudges. And since we're using a real tilt bob, we don't need the crude approximation of the tilt bob provided by that counter/timer system. Our *real* tilt bob already registered a real tilt, so we don't need the script to make any further decisions about it - we just want to send the tilt switch signal directly to the game's ROM. That's what would happen in the real version of the game, so it's what we want to happen in the simulation, too.

Here's how you can fix the shared script to accomplish this:

- For VP 10.4 and later:
 - Go to your Visual Pinball program folder
 - Open the **Scripts** folder you find there
 - Find the file **NudgePlugIn_mjrAccelAndTilt.vbs**, and rename it to **NudgePlugIn.vbs**
- For VP versions before 10.4:
 - Go to your Visual Pinball program folder
 - Open the **Tables** folder you find there
 - Download NudgePlugIn.vbs and place it in the **Tables** folder (make sure the filename is **NudgePlugIn.vbs**)
- Double-check your keyboard encoder device setup to make sure that your tilt bob is set to generate **T** key presses
- In Visual Pinball, go to Settings > Keyboard, and make sure that the keyboard key for "Mech Tilt" is set to "T". That's the default, but it's worth checking that it didn't get changed accidentally.

If you read through Chapter 15, Pinball Software Setup, you might recall my advice about setting up a text file where you record your customizations. This would be a good thing to add to that file now, so you'll remember it if you have to set up VP again at some point.

You should now have proper tilt bob handling for most tables. Virtually all modern tables with electronic displays should work with this, because the VP versions of those tables almost always use the shared scripts.

You'll probably run into some exceptions - tables that don't use the shared scripts, and so don't benefit from this custom version of the shared script. This is especially likely for older "EM" or "electro-mechanical" tables from the 1960s and earlier, the type with mechanical score reels. The VP versions of these older tables often use custom scripts for tilt handling, because the shared scripts are designed with more modern games in mind. The symptom you'll see in these older tables is that they behave in the weird "default" way described earlier. That is, when your tilt bob fires, you'll see a "fake nudge" in the game rather than a Tilt condition.

Whenever you run into an older table that behaves like this, you'll have to do some hand-editing of its script. That'll require a little sleuthing work on your part, since you'll find the right section in the table's custom script code and replace it by hand. Here's the basic procedure:

- Open the table in the VP editor
- Open the script window (View > Script in VP 10, Edit > Script in VP 9)
- Search for the text "KeyDown". You should find some code that looks something like this:

```
Sub Table1_KeyDown(ByVal keycode)
    if keycode = 20 Then
        Nudge 90,2
        TiltIt
    end if
End Sub
```

- If you can find code like that with a test for **if keycode = 20 then** or **if keycode = keyBangBack then**, then all you have to do is delete the line that

starts with "Nudge".

- If you can't find a line exactly like that, but you find similar lines with tests like **if keycode = CenterTiltKey**, you should simply add a new block of code like this just after the Sub line:

```
if keycode = 20 then
    TiltIt
end if
```

- This is where some sleuthing comes in. You'll have to replace the line that reads "TiltIt" in the example above with whatever the equivalent in the actual script is. Look for the code that handles the similar cases, such **if keycode = CenterTiltKey**, and copy what it does, *except* that you should omit any lines that start with **Nudge**.

If you can't find code that looks like this, or you can't make enough sense of the code to see how to make these changes, try asking in the forums. Lots of people on the forums are adept at coding these scripts, so someone should be able to help you figure out the necessary changes.

How to configure VP for a "virtual" tilt bob

What if you don't want to install a physical tilt bob, but you still want VP to detect tilt conditions when players get too aggressive with nudging? In this case, you can enable VP's simulated software tilt bob. The simulated tilt bob will monitor the accelerometer nudge input, and will generate a press of the space bar key when the simulated tilt bob swings too far.

We're talking about this situation:

- You're using an accelerometer
- **But** you're **not** using a physical tilt bob

This is really easy to configure, since the simulated tilt bob is a built-in feature in VP.

- Set up your accelerometer, and configure it in VP
- Follow the the procedure to configure VP for a mechanical tilt bob, even though you're not actually using one - this will make VP treat the software tilt bob the same way it would treat a mechanical one, which is just what we want when using an accelerometer
- Go to the VP **Keys** dialog, and check the box to enable the **Tilt sensitivity** setting

The last step is what enables the simulated tilt bob - the dialog refers to it rather obliquely as "Tilt Sensitivity".

The number in the Tilt Sensitivity setting lets you control how much nudging it takes to trigger a tilt condition. Higher numbers make it more sensitive. There's no rule for what this setting has to be - it's just a matter of experimenting with it to get the feel you prefer, by playing games and testing how much nudging it takes to trigger a tilt. If it feels too easy to trigger a tilt, increase the number.

Older nudging schemes

In the early days of virtual pinball cabinets, it wasn't as easy to find digital accelerometers as it is today. So early cab builders had to resort to other approaches

for nudging. There's no reason to think about any of these for a new build: digital accelerometers are simply the right tool for the job, plus they're cheap and easy to use. But for the sake of historical interest, we'll survey the schemes that older cabinet builders used.

Buttons. It's not exactly subtle, but one way of telling the machine you want to nudge it is to provide a button that inputs a "nudge" command to the software.

This is a direct carry-over from desktop pinball games, where you nudge by pressing a keyboard key (usually the space bar). Early cabinet builders just ported this idea to the cabinet by including nudge buttons.

Button nudging is simple, but it's not very satisfying in a cabinet, because it's not anything like how you play real pinball. So cab builders started looking for ways to detect cabinet motion.

Mercury switches. It's not common these days, but at one time there was a popular kind of light switch (the kind on the wall in your house) that had a little capsule of liquid mercury inside. The switch wires stuck into the capsule at one end. Moving the switch lever up would tilt the capsule so that the mercury ran to the end with the wires. Mercury is of course a conductive metal, so when the mercury spilled over the wires, it closed the connection and turned on the light. Moving the switch lever down made the mercury run to the other end, away from the wires, breaking the connection and turning off the light.

The point of these switches was to be quieter than regular mechanical light switches, since there was nothing inside to go CLICK. But cabinet designers realized they had another use. Because of the flowing mercury inside, you can use them as simple motion detectors. If you position one of these switches with the mercury capsule almost horizontal, with just at a slight tilt, a little push will send the mercury inside sloshing and make a momentary switch connection. If you wire one of these to the nudge key input on the PC, you can simulate a "nudge" key press by giving the cabinet a shove.

This approach eliminates the need for pressing buttons, so it acts a bit more like real nudging. But it's still pretty crude in that it can't detect how hard each nudge is. It's purely binary: nudge or no nudge.

Weighted joysticks. What we're really after is a way to detect not just when the cabinet is nudged, but how hard the nudge is. One way to do this is to use a joystick as a pendulum: hang the joystick upside down, so that the stick is pointing straight down. Put a weight on the end of the stick to give it some inertia. When you nudge the machine, the inertia of the weight will make the stick want to stay in place, which means that it appears to move in the opposite direction of the nudge, relative to the machine. By reading the joystick position, we can see how far it moves from center, which is a rough analog to the amount of force in the nudge.

This gives us the comparative strength of the nudge that we're after, so it's the best idea yet. But it's a fairly complex mechanical system. Most people who set these up find it difficult to get them to behave consistently. It's hard to keep the joystick precisely centered when everything's at rest, and the weight tends to swing back and forth after a nudge, which can generate spurious aftershocks. It's also hard to control the sensitivity, since the spring force and damping friction in the joystick aren't usually adjustable.

Old video game motion controllers. The next better approach is to use a motion controller from an old video game. These generally look like joysticks to the PC, and inside they have an electronic accelerometer that senses when the controller is tilted or moved. This is very similar in principle to the weighted joystick, but it's easier to

set up mechanically.

This approach came closest to the modern accelerometer solution. The downsides are that you had to find an old video game to scavenge, and that the accelerometers in these old controllers weren't very good by modern standards. In addition, many were designed for proprietary video game consoles rather than PCs, so you need additional software to make them emulate joysticks. More software, more problems.

1 **Some technical details on why Windows joystick calibration is bad**

for accelerometers. The purpose of the Windows joystick calibration is to normalize the input range of a mechanical joystick so that it matches Windows's internal definition of the range. Joysticks send position data to Windows saying how far left-to-right they are and how far front-to-back they are. The joystick defines the range of those readings in whatever quirky unit system it wants to use. For example, it might say that fully left equals -1000 units and fully right equals +1000 units. But many mechanical joysticks can't actually reach the limits of their defined ranges, simply because the stick hits the physical stops before getting all the way to +/-1000 units. The point of the Windows calibration is to measure the *actual* range that the mechanical stick can traverse, by asking you to move your joystick to each limit point and observing the reading. Windows then stores those min/max measurements, and applies a correction factor to all subsequent readings so that whatever reading the joystick reported at the maximum position is translated into the nominal -1000/+1000 maximum point of the joystick's desired unit system. This works great with real joysticks, but it's both unnecessary and harmful for accelerometers. It's unnecessary because accelerometers are pre-calibrated to report physically accurate numbers; applying any "correction" factor to a number that's already physically accurate will only make the number less accurate. Windows calibration is actively harmful for accelerometers because it's more or less impossible to give the calibration tool an accurate full-scale reading, which is the key piece of data that the calibration tool collects. There's no good way with an accelerometer to apply a smooth, steady, and accurate full-scale deflection for long enough that the calibration tool can accurately read it. (Well, there is one way: for an accelerometer whose full scale is +/- 1g, you can use the Earth's gravity to apply a 1g acceleration to one axis at a time, but only if you can hold the accelerometer perfectly still, with that particular axis pointing straight up and down.) The resulting bad data that the calibration tool collects will be applied as bad normalization factors to every subsequent reading, which will distort the VP nudge input. The problem is even worse than it appears, because Windows applies the normalization factor separately to the positive and negative side of each axis - so the "corrections" won't only be inaccurate, but they'll also be lopsided. That'll make left/right and front/back nudges weirdly asymmetric and non-linear.

37. Plunger

The plunger is certainly one of the defining features of pinball. It's as iconic as the flippers and steel balls. It's also one of the trickier parts of building a virtual cab. Plungers obviously aren't standard input devices for PCs, like keyboards and mice. There really isn't anything in the standard set of PC input devices that's at all equivalent. If you want a plunger for your cab, you need specialized hardware, purpose-built for the job.

Happily for us cab builders, such specialized hardware exists. There are a couple of commercial options available, and the Pinscape software offers multiple sensor options that you can build yourself.

In this chapter, we'll look at the plunger hardware options available, and some of the issues involved in planning and installing them. The good news is that there are lots of good options if you want a plunger, but all of them require some planning work.

Plunger or Launch button?

The first decision you should make about plungers is whether you want one at all. Most real pinball machines have plungers, but a plunger isn't an absolute necessity for a virtual cab. A simple Launch Ball button is adequate - although a compromise, because it doesn't give you the same degree of control as a physical plunger.

There are really three configurations to choose from:

- Plunger *and* launch button. The most popular option among pin cab builders. Lets you choose the best type of control for each game.
- Launch button only. Simpler and cheaper, but sacrifices the control of a real plunger for skill shots.
- Plunger only. This is an option if you're using Pinscape or a Zeb's Boards plunger, because these devices can do double duty as virtual Launch buttons when needed.

Option 1: Plunger *and* Launch button

I haven't done a scientific survey of cab builders, but I think the most common answer to "plunger or launch button?" is "both". With the plunger in the standard position, there's room directly below it for a Launch button, at least in a full-sized cab.

Some cab builders flip this arrangement upside down, with the Launch Button in the normal plunger location and the plunger located below it. I don't personally like the appearance of that; it always looks to me like the plunger got installed in the wrong spot. If part of your motivation for installing a plunger is a more realistic look, this works against that purpose. Aesthetics



aside, though, there are some pragmatic advantages to the inverted arrangement. One is that it's arguably more ergonomic: the plunger-on-top arrangement makes the Launch Ball button a little inconvenient to reach, since the plunger sticks out enough to be in the way. The other is that it gets the plunger out of the plane of the TV, which makes it possible to position the TV all the way forward against the front wall of the cab. If you were going to move the plunger down from its normal spot anyway to accommodate a TV, this arrangement can at least get some extra mileage out of that.

Option 2: Launch button only

Some pin cab builders choose to forego a plunger entirely and just use a Launch Ball button. This is a perfectly viable option functionally, because all of the PC pinball programs let you operate the on-screen plunger via the keyboard. And it doesn't make your machine look unrealistic, since a number of popular real arcade pinball machines also used button launchers.

If you're accustomed to playing pinball on your desktop PC, you already know how button-based plunging works. The convention on most desktop pinball games is that you press and hold the Enter key to start pulling back the plunger. As long as you hold down the Enter key, the plunger keeps retracting at steady pace. As soon as you release the Enter key, the game fires the plunger from however far back it was at the the moment of release. You control the strength of the launch by timing the release. To make this work with a pin cab, you just wire the Launch Ball button to act like the Enter key.



This time-based plunger action is a compromise for playability, obviously. It doesn't give you the same control you'd have with a physical plunger, and it's not a very good translation of the mechanics of a real plunger. But it at least serves the function, letting you make do with just a button if that's your preference.

Option 3: Plunger only

If you use a Pinscape Controller plunger or one of the plunger kits from Zeb's Boards, it can serve double duty as a Launch Ball button for tables that don't have conventional plungers, such as *Medieval Madness* or *Terminator 2: Judgment Day*. This makes it possible to play every type of game using just the plunger, so that you don't have to install a separate Launch button.

If you're using another plunger device, you should stick with the "plunger and Launch button" option. The ability to simulate the launch button is a feature of the plunger device, and other devices besides Pinscape and Zeb's don't generally have this ability.

If you're building a Pinscape Controller plunger, see Chapter 109, ZB Launch Ball for details on how to set this up. If you're using a Zeb's Boards unit, consult the owner's manual for the device.

Which option is best for you?

The only reasons I'd consider *not* including a plunger in your cab is if you're on a very tight budget, or you want to keep the project very simple. A plunger has such strong advantages that I'd only decide against one because of some external constraint like that. In terms of functionality and aesthetics, I think there's no contest.

If you do decide on a plunger, I personally think it's best to include a Launch button as well, so that you can use the control type that exact fits each game. But if your plunger can simulate a Launch button (as Pinscape and Zeb's Boards plungers can), this is only a slight edge functionally. You can choose according to whether you think more buttons will make your cab look cooler or make it look cheesier.

Why do I give the plunger such high priority? There are two main reasons. The first is aesthetics: a plunger makes your pin cab look more like a real pinball machine. The plunger is practically a defining feature of pinball, so its presence will instantly convey to anyone looking at your cab that it's a pinball machine. Yes, there are some real arcade pinballs that use launch buttons or gun triggers or something else in place of a plunger, but to some extent they do that to stand out from the crowd. With pin cabs the challenge is to make them stand out less from the real machines than they already do by virtue of their virtual-ness.

The second reason is that plungers are actually useful for game play. If you're not an experienced pinball player, plungers might seem pretty binary: you pull it back and let it go. But if you've played more seriously, you know about the venerable tradition of the *skill shot*, an element of many tables where you can score a bonus by launching the ball with just the right speed or timing. That requires precise control. A good virtual plunger can give you that control; a button just can't. The extra control adds to the fun for games with skill shots.

To summarize, here are the advantages of each option:

Plunger	Launch Button
Realistic	Simple
Skill shots	Cheap
Classic look	Modern look

Choosing a plunger device

This can be a tough decision. There are several options available, with different tradeoffs. Let's look at what's available and the relative advantages of each option. We'll start with a quick comparison chart for easy reference, then go into the details on each of the options.

The prices shown are only estimates, and of course they're likely to change over time, as prices tend to do! For the DIY options, the estimates are even more approximate, since there are different ways you can build the projects. For example, many of the DIY plans include custom 3D-printed parts. If you have your own 3D printer at home, you can fabricate those for the cost of the filament, which might only be a few dollars; but the same parts might cost \$10 or \$15 if you have to order them from a commercial 3D print service. And for that matter, you can sometimes make do without the 3D-printed parts, by substituting something improvised. To be conservative, though, my estimates assume that you're using the recommended 3D-

printed parts and ordering them through a commercial print service.

Also note that the Pinscape options all require a KL25Z microcontroller board to run the Pinscape software. I didn't include the price of the KL25Z in the price estimates, since I'm assuming that you're already pricing that into your system for its other features. A single KL25Z can handle the plunger along with all of the other Pinscape functions, so you just need the one. If you *weren't* already planning to include a KL25Z in your system anyway, you should add \$15 for the KL25Z to the price estimates for the Pinscape plunger options.

Device	Type	Price (est.)	Degree of difficulty	Accuracy	Features
VirtuaPin plunger kit	Commercial	\$160	Easiest	Not tested	15 buttons, accelerometer
Zeb's Boards plunger kit	Commercial	\$150	Easiest	Not tested, est. Very Good (<1mm)	19 buttons, accelerometer
Pinscape potentiometer	Open source	\$20	Medium Low	Very Good (<1mm)	All Pinscape features
Pinscape with Oak Micros potentiometer	Open source	\$25	Low	Very Good (<1mm)	All Pinscape features
Pinscape TCD1103	Open source	\$50	High	Excellent (0.1mm)	All Pinscape features
Pinscape AEDR-8300	Open source	\$30	High	Excellent (0.1mm)	All Pinscape features
Pinscape VCNL4010	Open source	\$10	Low	Good (1mm)	All Pinscape features
Pinscape VL6180X	Open source	\$20	Medium	Low (1cm)	All Pinscape features
Pinscape TSL1410R	Open source	N/A	Medium High	Very good (0.25mm)	All Pinscape features

("Not tested" means that I don't have any hands-on experience with that device, so I can't say how well it works compared to the options that I've tried in person.)

Top picks: I've tried to provide all of the details to let you make a fully informed decision yourself, but if you want my summary opinion, here are my top picks according to what you consider the most important priority:

- If you want it super easy: the Zeb's Boards kit
- If you want it super cheap: Pinscape with potentiometer or VCNL4010
- If you want the Pinscape features, with fairly easy setup: Pinscape potentiometer, VCNL4010
- If you want the best performance, and you're up for a more challenging build:

Now let's look at the available options in depth.

Commercial options: The two available commercial options that I'm aware of are from VirtuaPin and Zeb's Boards. You might also see an old product called the Nanotech Mot-Ion Adapter mentioned in the forums, but that was discontinued years ago and is no longer available for purchase.

- VirtuaPin's product uses an IR proximity sensor to detect the plunger position. That's a nice design in principle because it's physically simple and reliable, but when I tested their version 2 product years ago, I found that it was too low-res for my needs. Their version 3 product (current as of 2021) uses a different sensor that has better native performance, but I haven't tested their implementation. The v2 sensor resolution was the only problem I had with this product, though; otherwise it's a solid offering, well packaged and easy to set up. It includes a button encoder that lets you connect about 15 cabinet buttons, and an accelerometer for analog nudge input. It's not expandable, though, and 15 buttons is a bit limiting for a decked-out pin cab.
- The Zeb's Boards plungers use slide potentiometers as sensors. I haven't tested any of Zeb's plunger products myself, but I consider the basic physical sensor type sound because the same sensor type works well with the Pinscape software. The Zeb's Boards products include button input connections and accelerometer nudging features similar to the VirtuaPin product. Zeb's has an excellent reputation for technical support.

The big advantage of the commercial products, and it really is a big advantage, is ease of setup. They both come as complete packages, with all necessary parts included, and installation is simple for both. They also come with official technical support from the vendors (with a personal touch, too, since both are small businesses; you won't have to talk to an outsourced tech support call center).

The downside of the pre-packaged products is that they're closed systems with somewhat limited feature sets. Not overly limited - they both have good coverage of the basic pin cab necessities. But the cost of keeping things simple is that they don't offer much configurability or flexibility to expand beyond their fixed features. And of course the software is proprietary, so there's no way to add features or fix bugs yourself; you have to rely on the vendors for bug fixes and feature upgrades.

Pinscape options: Pinscape is a semi-DIY option. "Semi" in that you have to buy all the parts and do all of the physical setup work yourself, but you don't have to figure everything out from scratch, and you don't have to write any of the software. This build guide has plans that you can follow for a number of plunger sensor options, including parts lists and assembly instructions, so while it's not as easy as ordering a finished product and plugging it in, it's also not a research project. And of course all of the software is already written. If a research project is what you're looking for, though, this can be a good starting point. It's all open-source, so you're free to modify the sensor designs and software if there are things about them you want to improve.

All of the Pinscape options naturally require the Pinscape software running on a KL25Z, so you should factor the cost of the KL25Z (about \$15) into the overall price if you weren't already planning on a Pinscape device anyway. (If you were, you won't need a separate KL25Z for the plunger - a single KL25Z can handle all of the Pinscape functions simultaneously.)

The Pinscape software has built-in support for the following sensor types:

- **Potentiometer:** A potentiometer is a variable resistor, in this case one with a sliding lever that smoothly varies the electrical resistance level as you move it from one end to the other. They make these primarily for audio mixing panels and similar control panels, but they also work well for plunger position sensing, since the plunger slides back and forth in a straight line. The plunger travel range is about 80mm long, so we need a sliding potentiometer with a slightly longer travel than this. Suitable pots around 100mm long are easy to find.

The performance of these sensors is pretty good. In my testing, you get accuracy of about 1mm, which is good enough for smooth on-screen animation and tracking. The only negative is that the analog nature of the device means that there can be some random noise in the signal, which shows up on-screen as "jitter". It's pretty minor, and the Pinscape software has a filtering option to reduce it, with some trade-off in accuracy.

Pots are the least expensive type of plunger sensor, since the only required part (other than KL25Z) is the potentiometer, which runs about \$6 at Mouser. If you want to keep it really cheap, you can improvise your own mounting apparatus out of plywood and generic fasteners (L-brackets or that sort of thing). If you want to make it a little tidier (at slightly higher cost), the plans in this guide include a 3D-printable mounting bracket that you can fabricate.

- **Oak Micros's potentiometer (no longer available):** This works the same as the DIY potentiometer above, but it comes with an easy-to-install mounting bracket and saves you the work of sourcing the parts and assembling and wiring everything. I don't think this is available any longer, because Oak Micros announced in June 2021 that they're no longer shipping any of their products. You can check the original announcement on [vpforums](#) to see if there are any updates:

Announcement: Oak Micros Plunger Kit Mk II

- **VCNL4010:** An IR proximity sensor that can measure the distance to a nearby object, such as the end of the plunger. For a Pinscape setup, you mount the sensor near the end of the plunger, so that it can measure the distance between itself and the plunger; that serves as a measurement of the plunger's current position. This sensor is cheap (about \$7.50) and easy to set up (maybe even easier than the potentiometer), and it's completely non-contact (no wear and tear from moving parts). Its performance isn't quite as good as some of the other options (the potentiometer, AEDR-8300, and TCD1103 are all more precise), but it's still pretty good. Given its low price and easy installation, it's worth considering.
- **TCD1103:** This is an optical imaging sensor that detects the plunger position by rapidly taking pictures of the plunger. The TCD1103 chip is a high-resolution CCD (a type of camera sensor) that produces great quality images, which makes for excellent performance in the position sensing. It's capable of reading the plunger position to better than 1/300", with great stability, which makes for a very smooth on-screen response. The downsides of this sensor are that it's complex to build, and fairly expensive. It requires some additional electronics to interface to the KL25Z, as well as a small lens to focus the plunger image onto the sensor. I've designed a printed circuit board and 3D-printable mounting bracket (both open-source, of course) to bring it all together, so it's fairly straightforward to assemble one of these systems using my plans. However, there are enough parts involved that it does take a little online shopping work to source everything - plus, the printed circuit board uses SMD (surface-mount) components, which can make the soldering job intimidating if you haven't worked with these before. But I really like this sensor for its

excellent performance and the fact that it has no mechanical contact with the plunger (so there are no moving parts to wear out). See Chapter 105, Plunger Setup (TCD1103).

This is a fairly expensive option. The sensor chip all by itself costs about \$15 at Mouser, and you'll also need a lens (around \$8), lens holder (about \$1), circuit board (\$6-\$15), a 3D-printable mounting bracket (around \$10 if you have to order it from a commercial 3D print service), and a few other electronic parts (\$5), for a total around \$50.

- AEDR-8300: This is a specialized IC chip known as a "quadrature encoder", which uses optics to detect motion across a pattern of uniformly spaced, alternating black and white bars. The sensor tracks motion by counting the bars it passes. The bars are closely spaced, 75 line pairs per inch, and the sensor can determine its position to half the width of a bar, so the position reading is accurate to 1/300". This is an excellent sensor in terms of accuracy and stability; when set up properly, it really does achieve that 1/300" accuracy, which makes for silky smooth animation and tracking in the on-screen plunger. The big downside is that it's rather complex to set up, both because it requires a bunch of specialized (but easily fabricated) parts, and because the AEDR-8300 chip itself is a tiny SMD (surface-mount) chip that can be intimidating to work with if you haven't done SMD soldering work before. See Chapter 104, Plunger Setup (AEDR-8300 Encoder).

This option requires a custom circuit board (\$5), the electronics for it (\$10), a laser-cut acrylic piece (about \$1, although it's only practical to order in quantities of about a dozen), and a 3D-printable bracket (\$15), which adds up to about \$30 in parts.

- VL6180X: This is a "time-of-flight" IR distance sensor, which means that it measures the distance between the sensor and a nearby object by measuring the amount of time it takes for a pulse of light to reflect off of the object and return to the detector. For use with a plunger, you position the sensor at the end of a tube that you place around the plunger, with the sensor pointing at the plunger tip to measure the distance to the tip. The software works out the plunger position using the distance reading, knowing that the sensor is always at the same fixed position. These are relatively cheap and very easy to set up, since you can buy pre-built boards featuring these sensors from several hobby-electronics companies. Unfortunately, I don't consider these accurate enough to be usable - close, but not close enough. They nominally take distance readings in 1mm increments, but they're really only accurate to about 1cm. That makes the on-screen plunger animation very "chunky" when you connect them to a pinball program. They need about 10x better accuracy to be really workable; maybe the next generation in a few years will achieve that. I don't recommend these, but given how easy they are to set up, some people might find them "good enough." See Chapter 106, Plunger Setup (VL6180X Distance Sensor) if you want to read more about these.

You can buy pre-built boards with this sensor for about \$15 from Sparkfun, Adafruit, and some other hobby robotics companies. You'll also need to improvise some kind of mounting bracket, which might add a little cost if you come up with something requiring 3D printing.

- TSL1410R/1412S: This is the late, great, original Pinscape sensor, but sadly, the manufacturer stopped making it and the supply dried up a long time ago. This sensor was a linear photosensor array, consisting of a single row of 1280 pixels (1410R) or 1536 pixels (1412S). The row of pixels was by a magical coincidence roughly the same length as the overall plunger travel distance of

about 80mm, so the idea was that you placed the sensor near and parallel to the plunger rod, and placed a light source on the other side; the software read the position by taking a snapshot of the pixels and scanning the image for the shadow cast by the plunger. This worked pretty darn well and was only middlingly difficult to set up, although the sensors themselves were fairly pricey (about \$40). But alas, it's more or less impossible to build this design now since it's more or less impossible to find the sensors. But for the sake of historical reference, you can still read about it here: Chapter 101, Plunger Setup (TSL1410R Optical Sensor).

The sensors listed above are the ones that are already supported in the software. But they're not the absolute last word in sensors by any means. It's perfectly possible to add new sensor types, if you come up with something not already supported. The software internally uses an abstract C++ class for the basic plunger interface; each actual sensor's code is written as a subclass of this abstract base class. Adding a new sensor is a matter of adding a new subclass. You can do that yourself through the miracle of open-source software, or you might well be able to persuade me to write the code if you come up with something that improves on the sensors already supported.

Fully DIY options: It's certainly possible to come up with a whole new design of your own, without any commercial products involved and without basing anything on the Pinscape software or hardware plans. I don't think full DIY is the best option for most people, given that the Pinscape software is open-source, meaning that can use it as a starting point no matter how radically you want to change or customize it. That should save you a ton of time compared to starting completely from scratch. On the other hand, if you're as fond of tinkering with these things as I am, the challenge of building a whole new system from scratch might be way more appealing than just adapting an existing piece of software.

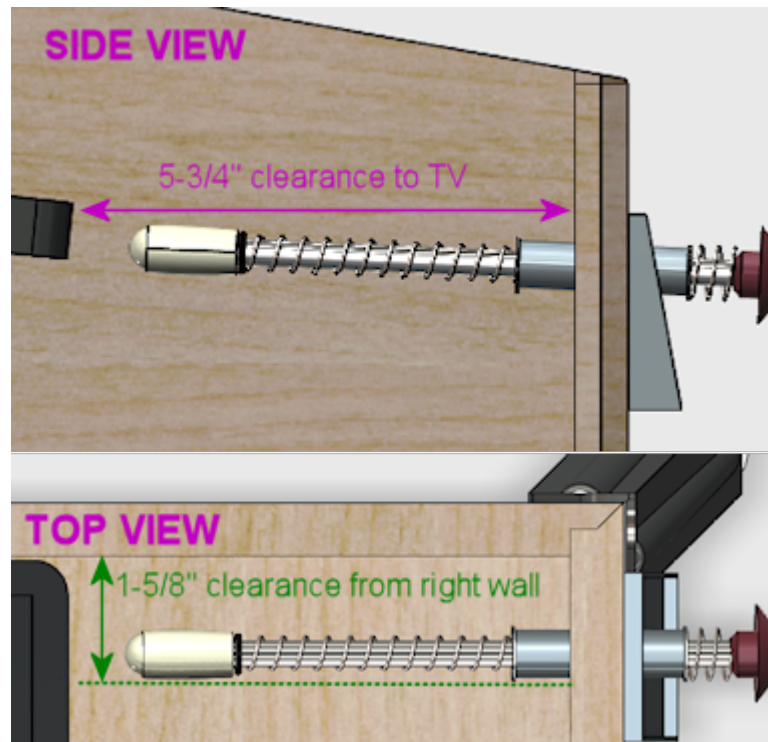
It would obviously defeat the purpose of "fully DIY" to give you a list of particular DIY options here. But purely to spark your imagination, I'll mention some approaches I've heard about, without going into too much detail:

- Use a computer mouse that's either attached to the plunger rod, or positioned so that it can scan something attached to the plunger rod. See "My Mouse Plunger Setup (aka... Cheap :)": www.vpforums.org/index.php?showtopic=38064.
- Some early pin cab builders created a sort of hybrid of the plunger and launch button by using a microswitch at the end of the plunger travel as the sensor. The switch was connected to a button encoder as the Enter key, so that pulling back the plunger by any amount acted like pressing Enter, and releasing it would hit the switch again and release the Enter key. You launched the ball using the desktop convention of a timed plunger pull based on how long you held down the Enter key. This isn't a position sensor by any means, but it's simple and at least creates the appearance of a plunger.
- One person on the forums several years ago used an LVDT (linar variable differential transformer), a type of position sensor that uses inductive coils to sense the position of a metal rod. Sounds perfect for a plunger sensor, doesn't it? The snag is LVDTs are super expensive (hundreds or even thousands of dollars) and hard to come by. LVDTs were apparently popular in industrial applications ten or twenty years ago, but they seem to have been largely replaced by optical and magnetic quadrature sensors in more recent times. The ones still on the market are ridiculously expensive specialty products that are way out of range for a pin cab project. It also looks like the electronics to interface one to a microcontroller are pretty complex.

Positioning the plunger

Before you start drilling holes for your plunger, you should carefully consider all of the other things that have to fit into the same area, to be sure you don't have any conflicts when you start installing things.

A standard plunger sticks into the cabinet by about $5\frac{3}{4}$ " from the inside of the front wall. It occupies the area out to a minimum of about $1\frac{5}{8}$ " from the inside right wall for the plunger rod itself, but your plunger sensor might require extra clearance on top of that. For example, the Pinscape AEDR-8300 sensor requires a plastic part to be attached to the plunger rod, which increases the clearance area to about $1\frac{3}{4}$ " from the inside right wall.



Clearances required around the plunger.

Things to take into account when determining the plunger position:

- The TV. On a real pinball machine, the plunger is in roughly the same plane vertically as the playfield. They make this fit on a real machine by cutting a plunger-sized notch out of the playfield at that corner. That's not helpful for virtual cabs because you can't cut a notch out of the TV. If you're positioning the TV in the same plane as the plunger, you'll have to leave a gap between the front of the cabinet and the front of the TV to make room for the plunger. If you don't want to leave a gap, you'll have to move the plunger down far enough to get it out of the way of the TV.

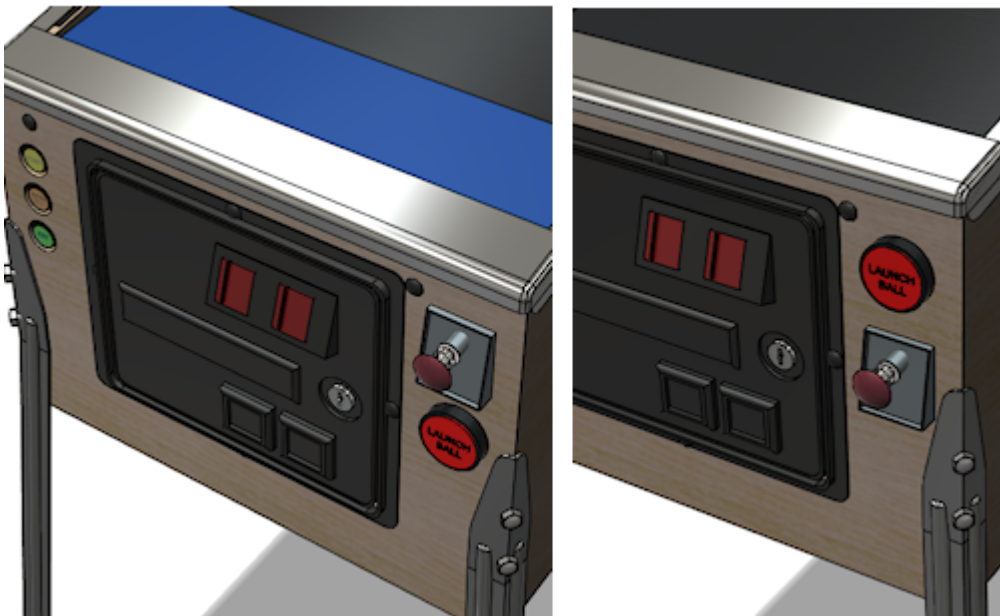
See also "The dreaded plunger space conflict" in Chapter 29, Playfield TV Mounting.

- The right wall of the cabinet. This constrains how far to the right the plunger can go. The standard plunger position, shown in the drilling template below, is positioned about as far to the right as it can possibly go, due to the bolts on the housing. Don't try to move it further right.
- The flipper buttons. The plunger is usually in the same plane as the flipper buttons. If you're using standard leaf switches like on a real machine, there should automatically be enough room, since the "right wall" constraint above

leaves enough room (just barely) for the buttons and switches on a real machine.

- The coin door. This limits how far left you can move the plunger. On a standard-width cabinet, there's about 1" of clearance between the plunger housing and the coin door. (You'll have more clearance if you're building a widebody cabinet or a custom size that's wider than standard.) This isn't usually a significant constraint since you usually want the plunger positioned as far right as it can go.
- The right front leg. If you're using real pinball parts for the legs, this constrains how far down you can move the plunger from the standard position. You can move down by at most about 3" from the standard position.

Plunger on top vs. Launch button on top: Most cab builders who include both a plunger and a Launch button put the plunger on top. But some people invert the stacking, placing the Launch button in the normal plunger spot and moving the plunger down a few inches.



Above left: Exterior appearance with the standard plunger placement, with plunger on top and Launch button below. Above right: Inverted arrangement with the plunger on the bottom.

The main reason to put the plunger on the bottom is to make room for the TV to come all the way to the front of the cabinet. With the plunger on top, you'll probably have to push the TV back a few inches to leave enough room for the plunger; some people hate the idea of that gap between the TV and the front of the cabinet. I personally find the gap benign, and in fact I even prefer a little set-back, so that you're not looking straight down at the flippers. The inverted arrangement also looks weird to my eye, since the plunger is *always* at the same spot in the real machines.

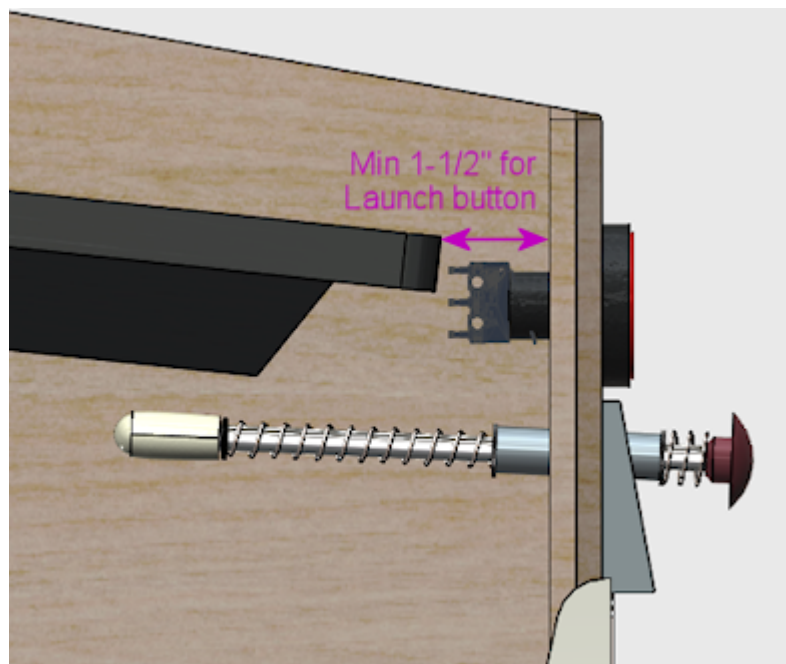




Lowering the plunger to get it out of the way of the TV, so that the TV can be moved all the way forward to the front of the cabinet.

If you decide to move the plunger down to make room for the TV, be sure to measure everything carefully with your actual TV. The shape of your TV case is important here, since that determines how far down you'll have to move the plunger to clear the back of the case. Also pay attention to the slight upward tilt of the plunger rod relative to the housing. The rod is angled upward at about 3°, which makes the front of the rod slightly higher than the holes drilled in the front wall (see the diagram below). The open area needs to be about ¼" higher than the top of the drilled holes in the front wall.

If you're including a Launch Ball button in the position where the plunger normally goes, make sure you leave room for its intrusion on the inside of the cabinet when positioning the TV. It requires about 1½" clearance from the inside front wall. This usually isn't a problem, because you'll probably want to position the TV at least 2" from the inside front wall anyway, since a standard lockdown bar covers up about that much space.



Drilling the holes

See "Plunger and Launch button" in Chapter 21, Cabinet Body for a drilling template for the plunger opening, and measurements for the standard placement of the plunger and Launch button. Remember to make any adjustments to those plans if you're repositioning the plunger vertically.

Standard ball shooter hardware

If you buy a commercial plunger kit, the plunger assembly is usually included in the price. If you're building one yourself, here are the parts you need.

You can buy fully assembled ball shooters from any pinball parts supplier, such as Pinball Life or Marco Specialties (see Chapter 4, Resources). Nearly all machines

made since about 1980 use the same assembly, which you can find listed at the pinball parts vendors under these Williams/Bally part numbers: B-12445-1, B-12445-6, B-12445-7.

Alternatively, you can buy the individual parts separately, if you wish to customize anything. Pinball Life lets you choose colors for the knob and rubber tip, but you'll have to buy *à la carte* if you want a custom knob. You can also buy a "knobless" shooter rod, which lets you create your own custom knob for a unique look.

Springs are available in different tensions. I'd recommend a lower tension spring for virtual pinball use, because you're never going to hit an actual ball. The energy has to go somewhere when there's no ball to hit, so it usually goes into rattling the cabinet. Lower spring tension reduces the speed and cuts down a bit on the rattling.

Here are the individual parts, with Williams/Bally part number references:

- Shooter rod: 20-9253
- Shooter housing: 21-6645-1
- Shooter housing sleeve: 03-7357
- Barrel spring ($\frac{3}{4}$ " long x $\frac{5}{8}$ " diam): 10-149
- Inner spring ($5\frac{1}{2}$ " long x $\frac{1}{2}$ " diam): 10-148-1
- E-clip ($\frac{3}{8}$ " shaft, $\frac{5}{16}$ " groove): 20-8712-37
- Washers ($25/64$ " x $\frac{5}{8}$ ", 16 gauge, qty 2): 4700-00051-00
- Rubber Tip: 545-5276-00

There's also a special mounting plate that goes with the ball assembly, which for some reason is never included in any of the complete assemblies or pin cab kits. It's not an absolute requirement, but it makes the installation easier and cleaner. You'll also need some specific machine screws, which also aren't included in the assemblies or kits; they're common parts you can easily find at a hardware store.

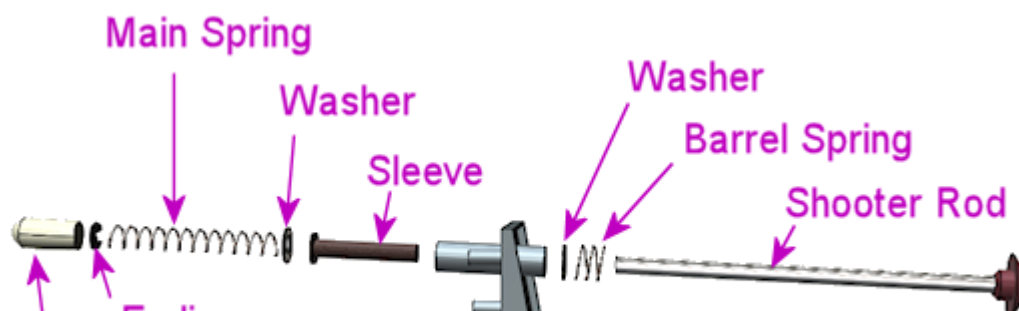
- Ball shooter mounting plate: Williams/Bally 01-3535
- #10-32 x $\frac{5}{8}$ " machine screws (quantity 3; $\frac{3}{4}$ " length will also work)

Custom knobs: Pinball Life sells a "knobless" shooter rod, which gives you the option to create your own completely custom knob. Use your 3D printer to create something unique. Fasten it with epoxy.

- www.pinballlife.com/index.php?p=product&id=1608

Custom knobs are popular "mods" for real machines. You can find lots of after-market options on the Web by searching for "custom pinball shooter". These will work just as well for virtual cabs.

How to assemble a standard plunger





Assemble the parts in the order shown in the diagram above:

- Slip the barrel spring over the shooter rod and push to the knob end
- Slip the washer over the shooter rod and push down to the barrel spring
- Insert the nylon sleeve into the shooter rod opening in the housing (from the inside of the housing)
- Insert the shooter rod into the opening the housing (from the outside of the housing)
- Slip the other washer onto the shooter rod
- Slip the main spring onto the shooter rod
- Attach the E-clip to the rod. You'll have to hold the spring back while you do this, since the spring will be compressed in its normal position. The E-clip fits into the groove near the end of the rod. Use needle-nosed pliers to snap it into position.
- Fit the rubber tip over the end of the rod. (This is optional in a virtual cab; you probably don't need the tip unless you're using some kind of optical sensor that requires it. Leaving it out will save a little space if you have tight clearance to the TV.)

How to install the ball shooter assembly

The plunger is designed to be fully assembled before you install it, so start by assembling the parts as described above.

Insert the housing into the drilled opening in your front panel, from the outside. Fit the plunger mounting plate over the screw holes in the assembly on the inside wall of the cabinet. Fasten with three #10-32 x $\frac{5}{8}$ machine screws. Make the screws fairly tight, since the plunger is subject to a lot of mechanical force when you use it (but don't overdo it - you don't want to strip the threads in the housing).

38. Plunger Setup on the PC

Most of the popular pinball player programs have support for plunger devices built in, but they all require you to configure some option settings to enable the plunger support and fill in the details of your device. This section explains how to set up Visual Pinball and Future Pinball, and how to do some simple tests to make sure everything's working properly.

Windows itself generally doesn't require any special setup to work with a plunger. Plungers emulate joysticks, and Windows has excellent plug-and-play support for joysticks built in.

Plunger device basics

Before getting into the details of setting up the software, it's worth understanding how a plunger device communicates with Windows and the pinball player software.

At the Windows device level, all of the plunger devices pretend to be joysticks. The Windows joystick interface defines several "axes" that report the positions of the moving parts of the joystick. A regular joystick typically has an X axis and Y axis that report the left/right and up/down position of the stick. These are reported numerically, on arbitrary scales; for example, the left/right axis might report position values from -1000 for "all the way left" to +1000 for "all the way right". That's just a made-up example; the actual numerical ranges depend on the individual device, and they're usually in abstract units that don't correspond to inches or centimeters or any other real units.

The convention used by all of the plunger devices I know of is to use the joystick "Z" axis to report the plunger position. For a real joystick, the Z axis is supposed to represent some kind of vertical motion of the stick, but that's just a convention; Windows doesn't know or care what the physical geometry of the device is like. If the device reports "Z=500", Windows just passes along "Z=500" to any application software (like VP) that asks.

As with all joystick axes, the numerical range of the Z axis is arbitrary and abstract. But Visual Pinball does define one special point on the Z axis: Z=0 (zero) is assumed to be the resting position of the plunger. That's the position where the plunger sits when it's in equilibrium, when it's not in motion and when you're not pulling it back. If you run through the plunger calibration procedure in the Pinscape Config Tool, one of its jobs is to identify the resting position and calibrate it so that it reports Z=0.

Visual Pinball and the other pinball programs that know about plunger devices all use this joystick interface. They look for a joystick device in the system, and if one is found, they read the Z axis value to determine the position of the physical plunger. Visual Pinball uses this to control the simulated on-screen plunger, so that the simulated plunger action in the game can be controlled by moving the physical plunger.

Plunger device axis setup

Some plunger devices give you options about which axis to use. The Z axis is the most common choice, since that's the one that almost all of the pinball player programs expect you to use. However, in some rare cases you might need to choose a different axis to avoid conflicts with other devices.

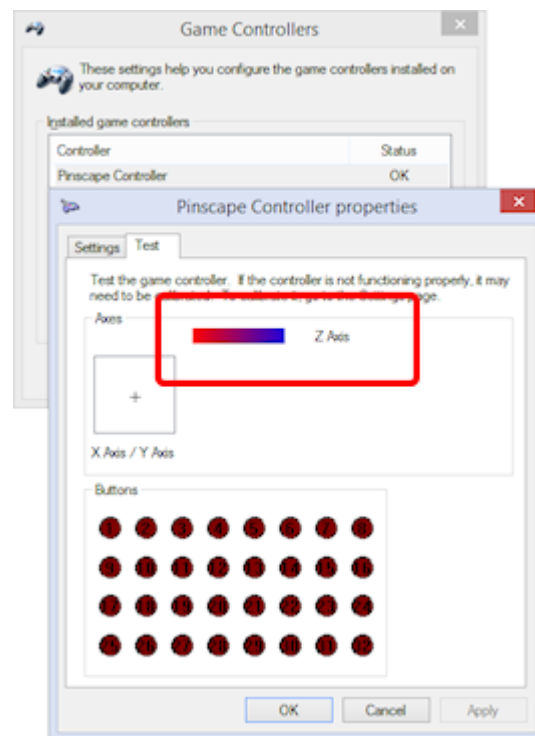
If your device does use some joystick axis other than the Z axis, be sure to

substitute that axis for the Z axis in all of the settings throughout the rest of this section.

Testing with the Windows joystick control panel

You can run a really simple test that your basic plunger hardware setup is working using the Windows joystick control panel. This is a good early test because it doesn't depend on your Visual Pinball or Future Pinball settings; it just tests the basic USB joystick input.

- Press Windows+R
- Type **joy.cpl** into the Run box
- In the Game Controllers list, double-click on your plunger device
- Look for the "Z Axis" bar on the properties page
- Move your plunger



If all is working, you should see the on-screen Z Axis bar track the motion of your physical plunger as you move it back and forth.

(If you set up your plunger on the Rz (rotational) axis, there should be a "Z Rotation" bar instead, and that should track the plunger motion.)

If the Z Axis (or Z Rotation) bar doesn't move, your plunger isn't sending USB readings to Windows properly. Go back and check your plunger hardware settings to make sure they're correct. If you're using Pinscape, you can use the Pinscape Config Tool's plunger tester to see the raw input from the plunger, to make sure that the position sensor is working.

If your plunger device doesn't appear in the Game Controllers list, make sure it's connected to USB and that it's configured properly in its hardware settings.

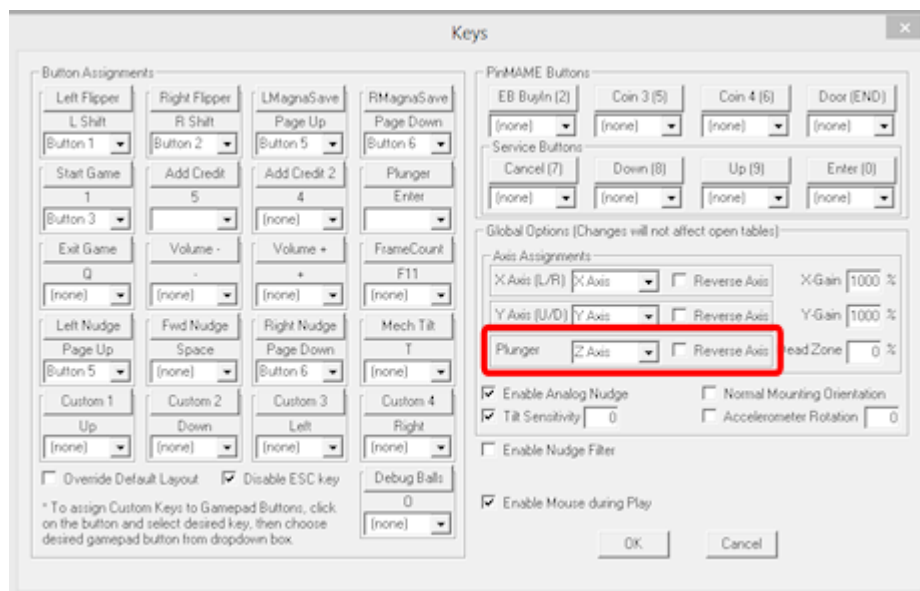
Important: do not calibrate! The Game Controllers control panel has a "Calibrate" button under the Settings tab that many people see and think it would be a good idea to click. Don't! This particular calibration process is designed for real physical

joysticks only. It's unsuitable for plunger/nudge devices and it'll make your device act erratically. If you clicked the Calibrate button at some point thinking it would improve matters, you should undo that, by returning to the Settings tab and clicking "Reset to default". That will erase the troublesome calibration data and restore normal operation. (This advice applies to the Pinscape Controller and most of the other nudge devices I know about, but there might be exceptions. If your device's owner's manual tells you to use the Windows joystick calibration despite what I just said, go with the owner's manual's advice.)

VP and FP plunger preference setup

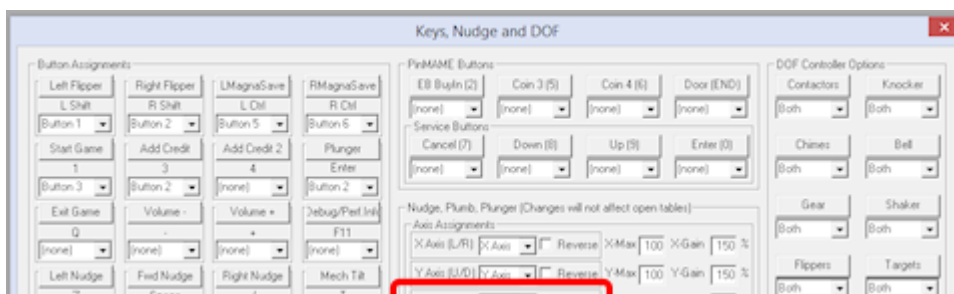
Most of the pinball player programs have option settings that let you specify whether or not a physical plunger is attached, and if so, how it should be read. If your plunger isn't working with a given pinball player program, the first thing to do is check the program's options and make sure the plunger is enabled and is using the correct joystick interface.

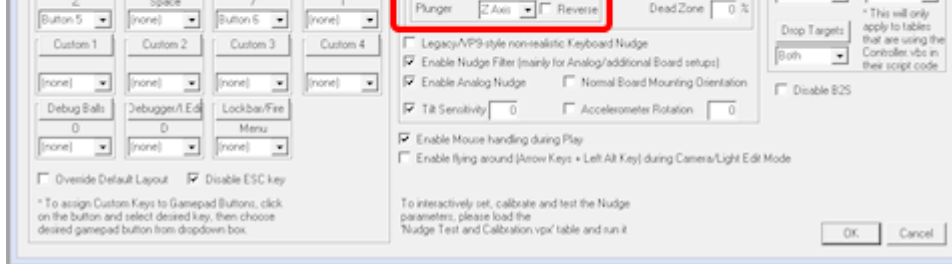
Visual Pinball 9: Open the VP editor. Select **Preferences > Keys** from the menu. Find the "Plunger" section. Set the plunger axis to match the one reported by your plunger device; this is usually the Z axis. If you changed your Pinscape settings to Rz (or you're using a different plunger controller that uses something other than the Z axis), select the appropriate axis here.



VP 9 plunger settings in the Keys preferences dialog

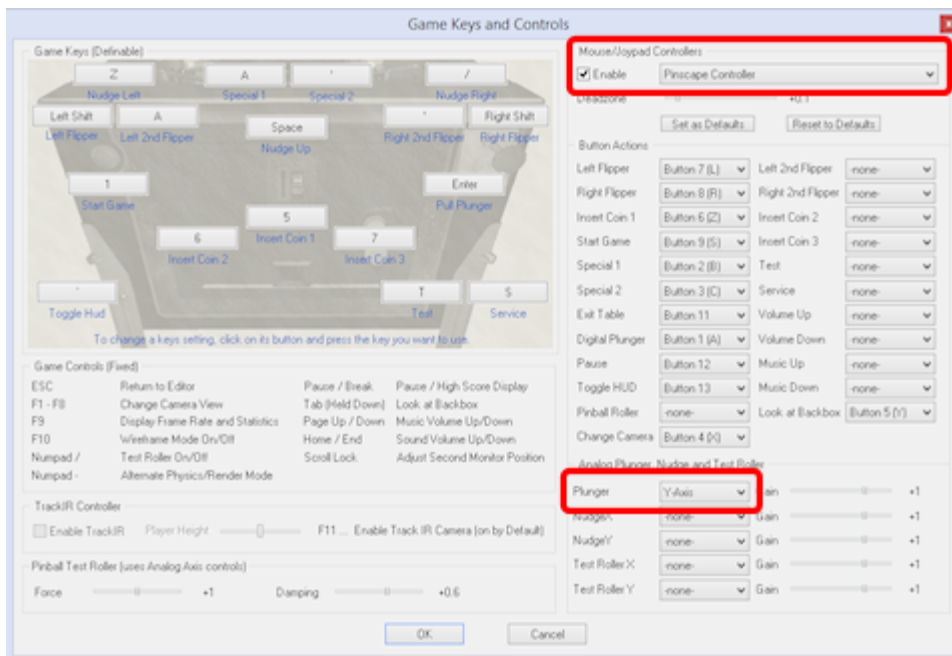
Visual Pinball 10: Open the VP editor. Select **Preferences > Keys, Nudge & DOF** from the menu. Find the "Plunger" section. Set the plunger axis to match the one reported by your plunger device; this is usually the Z axis. If you changed your Pinscape settings to Rz (or you're using a different plunger controller that uses something other than the Z axis), select the appropriate axis here.





VP 10 plunger settings in the Keys preferences dialog

Future Pinball: Open the Future Pinball editor. Select **Preferences > Game Keys and Controls** from the menu. Select your plunger device from the "Mouse/Joypad Controller" drop list, and make sure the "Enable" box is checked. Find the Plunger axis setting, and select "Z-Axis" from the drop list. (Or select the actual axis that your plunger device uses, if different.)



Future Pinball plunger settings in the keys & controls preferences dialog

Testing your plunger with VP or FP

It's a good idea to test a new plunger device with a stripped-down starter table in each player system, to make sure that the basic hardware and software setup is working. Once you know the plunger is working in a test table, you can be sure that any problems you have with individual tables are those tables' fault, not something wrong with your basic setup.

You should go through these steps with each player program that you'll be using, including each version of VP if you've installed more than one. The preference settings are specific to each program and version, so it's possible for VP 10 to be working but VP 9.9 to be broken, or vice versa.

Here are the basic steps to test the plunger device in VP:

- Run VP and go to the blank editor (cancel out of any initial "Open File") dialog
- On the menu, select **File > New** to create a new blank table
- Make sure the Properties panel is showing on the right; if it isn't, click the

Options button in the left tool window

- On the menu, select **Edit > Select Element**, then scroll down to the Plunger object and click on it
- In the Properties window, make sure **Enable Mechanical Plunger** is checked
- Press F5 to run the game
- Try moving your plunger

If everything's set up correctly, the on-screen plunger should track the motion of your physical plunger. If the on-screen plunger doesn't move, go back through the plunger preference setup steps above.

The procedure for Future Pinball is simpler, because FP doesn't have an equivalent of the "Enable Mechanical Plunger" checkbox that VP uses. It's just always enabled. So simply create a blank table and run it, and test that the on-screen plunger tracks the motion of your physical plunger.

Fixing individual tables

Okay, if you've made it this far, your plunger is sending joystick input to Windows successfully and is working with test tables in the pinball players you're using. Ideally, we'd be done at this point: just load up some tables and play.

There's just one slight problem with that: some individual tables might not work properly, particularly with Visual Pinball, and particularly with VP 9. This is despite the fact that we know that the plunger input is correctly reaching VP itself. The snag is that individual VP tables don't always use the built-in VP plunger object, or don't use it properly for real plunger input. Some tables use their own improvised scripting code that doesn't take plunger devices into account.

VP tables that don't work properly with a plunger device can usually be fixed, sometimes easily and sometimes with a bit of work. That's a fairly big subject, so we cover it separately in Chapter 39, Fixing VP Plungers.

39. Fixing VP Plungers

If you built one of the plunger sensor options for the Pinscape Controller, or if you bought a commercial plunger kit, you'll certainly want to use the plunger when playing simulated pinball tables. Many tables work with plungers automatically, but some don't.

Fortunately, for open systems like Visual Pinball, it's possible to edit the table files yourself, so you can often fix any games that don't already work with the plunger. This section shows you how to do this for Visual Pinball tables.

Plunger support has evolved in Visual Pinball over the years. Most Visual Pinball 10 tables should work automatically. It's much more hit-or-miss with Visual Pinball 9, though; many tables, especially older ones, lack proper support for the plunger.

Commercial game software doesn't usually give you any ability to modify the software yourself. Your only recourse for a commercial game that doesn't work is to contact the publisher's technical support people.

How plungers are supposed to work in Visual Pinball

The "right" way for a Visual Pinball 9 table to implement a plunger is to use VP's intrinsic plunger object. VP has had built-in support for plunger devices since early releases of VP 9.

The built-in plunger object in VP automatically takes input from a USB plunger device, if you have one in your system. This works by reading the USB joystick interface, using what's called the Z axis (although you can configure this in VP's settings dialog to use a different axis, if your physical plunger device uses something different). The name "Z axis" is taken from mathematics, where it conventionally refers to the vertical axis in a 3D plot. But PC joysticks use the term loosely, to refer to "just some random third axis" - *third* because the first two, X and Y, are the two degrees of freedom of the stick itself, left/right and forward/back. Z usually represents some other control attached to the joystick, such as a slider control or a throttle. For virtual pinball plungers, we only pretend to be a joystick, but we still have to use their notation. So we continue the PC joystick tradition of assigning idiosyncratic meanings to the mathematical axis names, using X and Y to represent accelerometer readings for nudging, and using Z to represent the plunger position.

Tables that take advantage of VP's built-in plunger feature usually "just work", because the built-in plunger itself just works (once you have it properly configured, at least).

Why plungers don't always work in VP tables

If tables authors would always use VP's built-in plunger object, plungers would work properly in all tables.

But table authors don't always go along with this plan. A lot of VP tables, especially older tables written before about 2016, *don't* use VP's built-in plunger feature. Instead, a lot of them reinvent the ~~wheel~~ plunger by using Visual Basic scripting (which is another thing built into VP) to create their own completely new object that's supposed to look and act like a plunger.

To be fair, many of the table authors that scripted their own custom plunger objects were well-meaning. The early versions of VP's plunger looked cartoonish and had a rather poor physics simulation. Table authors working on re-creations of real tables

naturally wanted their tables to look good and work right, so they didn't want to settle for the poor plunger implementation in early VP 9 releases.

But well-meaning or not, all of this re-inventing created huge hassles for those of us with physical plunger devices. Many of the table authors who created their own plungers in VB scripting didn't think to read the joystick input. If a table doesn't read the joystick input, it obviously won't work with your physical plunger device, which sends information via the joystick input.

VP's built-in plunger was overhauled around version 9.9.5 to give it a photo-realistic appearance and to make its physics work properly, both for desktop play using keyboard input, and for cabinet play with a physical plunger device. The new plunger was carried over into VP 10 and has been a part of VP 10 from the beginning. So newer tables - those written later in VP 9's tenure, and anything written for VP 10 - mostly use the new plunger object and mostly "just work". The only exceptions among newer tables are a few written by authors who were accustomed to the bad old way of doing things and didn't get the memo about the built-in plunger overhaul. But those should be increasingly rare as time goes on, and I certainly hope no one is still using the bad old way for new tables at this point.

How do you fix a table that ignores the joystick input? Unfortunately, it's not easy - but it can be done. Basically, you have to rip out all of the scripted plunger code in the table, and substitute the "real" built-in plunger object instead. The thing that makes this difficult is that there's not a simple, rote procedure you can follow, because every table that re-invented the plunger re-invented it in its own way. You have to look at each table individually to find its plunger scripting code and the associated table objects so you can remove all of that. But there are two rays of sunlight here. The first is that most table authors didn't actually reinvent their custom plungers from whole cloth - most did a copy-and-paste job from another table. So once you've fixed a few tables, you'll start recognizing common patterns, and it'll become easier each time. The second nice thing is that you don't really have to write any new code - mostly it's just a matter of deleting all of the custom plunger code so that the built-in plunger can take over. The table scripts will end up simpler and cleaner when you're done with them. At any rate, I'll try to give you the outline of a procedure that works for many tables, but be warned that it still takes some work on your part; it's not a simple "recipe" that you can apply mechanically.

Why plungers don't always work, part II

There's actually one more reason that plungers don't always work in VP. This one's a lot easier to fix than the first one.

Remember how I said that the built-in plunger in older versions of VP had bad physics? The situation was so bad that the bad physics actually interfered with ordinary desktop use, even if you didn't have a joystick attached. So a lot of table authors who used the built-in plunger as the basis of their scripted re-inventions *explicitly disabled the joystick input* so that desktop users wouldn't have to put up with the bugs.

So for some tables, you'll find that there's already a proper built-in plunger object present, and that all you have to do is tick a box in the settings to re-enable joystick input.

How to fix a VP table

Here's my procedure for fixing VP tables with broken plungers. Be warned that this

isn't a simple recipe you can follow mechanically; it's more of an outline.

Before starting, save a backup copy of the table! We're going to go into the table definition and do some major surgery. It's possible to hopelessly screw up the table this way, and sometimes the easiest way to get back to a working state is to scrap all of your changes and start over from the original working version. You'll feel better about messing around with the table's innards if you have a backup copy tucked away that you can easily restore if things go haywire.

If you're already adept at using VP, you probably just need a quick overview of the steps at a high level:

- First, check for a Plunger object in the table. If it's there:
 - Make sure **Enable mechanical plunger** is enabled in its properties
 - Make sure it's positioned so that it can hit the ball
 - Make sure it's visible
 - Set its style to Custom, and set the image to CustomWhiteTip or CustomBlackTip (see Making the built-in plunger look nice below)
- If there's not already a Plunger object, try adding one, setting it up to use the new "Custom" visual style and positioning it so that can hit the ball when fired
- Delete any playfield objects (ramps, walls, lights, EMReels) that are being used to draw the scripted plunger on-screen
- Go through the table's scripts and delete all of the old plunger-related scripting code

If you're not a VP power user, read on for a more detailed explanation.

Open the table in the VP editor

Fire up VP. Open the table in the editor (not in "playing" mode).

Look for an existing plunger object

The first step is always to find out if there's already a built-in plunger object. Many (maybe most) of the tables with scripted plungers *also* have a built-in plunger object that's used as a helper for some of the scripted operations.

If there is a built-in plunger, it might be hidden or located somewhere off the table or in a remove corner of the table where it can't interact with the ball. So it might not be obvious that it even exists when you look at the table layout in the editor.

The easiest way to find this object is as follows:

- Edit > Select Element (or Ctrl+Shift+E)
- Look through the list for something called "Plunger" or something similar, like "Plunger1"
- Click it in the list
- Click Select - a rectangular outline should light up somewhere in the layout showing where the object is positioned
- Close the dialog

Enable mechanical plunger

If you successfully found a plunger object in the previous step:

- Select the object as described above
- If the properties window isn't already showing, click Options in the left panel to bring it up
- In the State section, make sure the **Visible** box is checked
- In the State section, if **Park Position** is set to 0, change it to 0.16667
- In the State section, make sure the box for **Enable mechanical plunger** is checked
- If the **Enable mechanical plunger** box *wasn't* already checked, this might be all you need to do to fix it! This might be one of those tables where the original author disabled joystick input (this is the option that does that) because of the old physics bugs it caused for desktop users. Checking the box re-enables the joystick input, so the table might suddenly start working. Save the table, fire it up in "play" mode, and test it out. If the on-screen plunger tracks your physical plunger and launches the ball properly, you're done! If it *mostly* works at this point but the plunger is too weak or too strong, see "Other physics adjustments" below.

Try moving the plunger object

If you found a plunger object, but "Enable mechanical plunger" didn't turn out to be a magic one-step fix, there's another simple thing we can try before we have to dig deeper into the table.

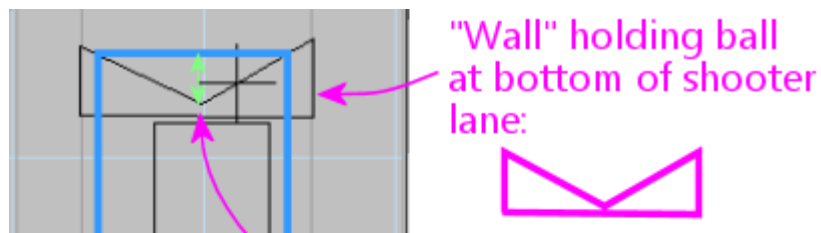
A lot of table authors who "re-invented" the plunger did so by tying their scripted object to a real plunger, and kept it from getting involved in the physics by moving it somewhere out of the way where it couldn't hit the ball. You might be able to see this easily just by looking at the table layout - the plunger might be obviously off in left field somewhere. But in some cases, this might not be visually apparent. Some table authors just move it back a few pixels from the ball position, so that it gets tantalizingly close but can't actually hit the ball.

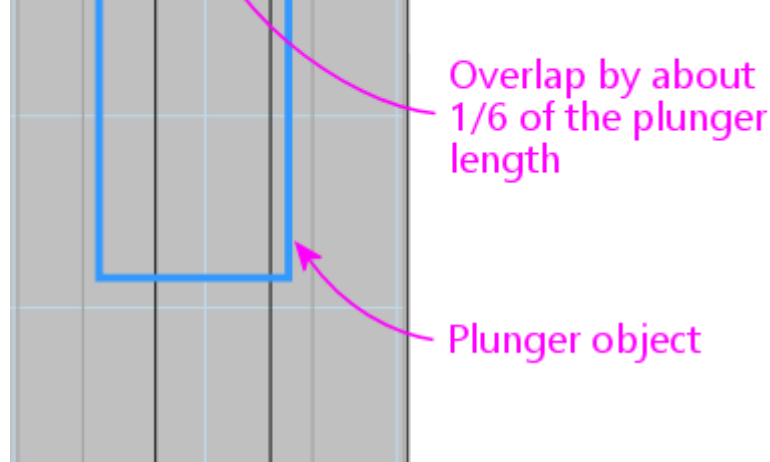
So the thing we can try now is moving it into the proper position.

The first thing to do is to make sure the object is movable. If the object lights up with a gray rectangle when selected, it's "locked", meaning VP won't let you move it. But we have the key! Right-click it and un-check "Locked" in the context menu. If it lights up in blue when selected, it's already unlocked and movable.

If the object is obviously off in left field somewhere, try moving it into the shooter lane area. If it's already more or less in the shooter area, try moving it further up on the playfield (in the upwards direction on the editor screen).

It can be hard to visually parse VP's editor screen, because everything is so schematic (just a few black lines showing the rough outline of each object) and there can be so many things overlapping in one area. Most games have an arrangement something like shown below, with a "wall" object shaped like a little wedge that holds the ball in place at the bottom of the shooter lane, just above where the plunger strikes. That's the key point for aligning the plunger. It has to overlap that wedge-shaped wall slightly - by about 1/6 of the plunger length - in order to hit the ball.





You won't find this exact layout on all tables, so don't be overly literal about looking for this exact picture. You might not find a "wedge" wall like this at all. It's a common motif, but every VP table author does things their own unique way. You might find a differently shaped wall in roughly the same area, or no wall at all.

There's also usually a bunch of other stuff overlapping in the same area, so even if the wedge is there, it might be hard to visually separate from all of the other objects. If you're having trouble identifying which squiggle belongs to which object, remember that you can select an object by clicking on it. This will at least highlight it so that you can see which of the lines belong to this one object, and it'll show its name and object type in the Properties window, which can help identify its purpose.

Once you've moved the plunger into what appears to be the proper position, test again in "play" mode. Again, check if the on-screen plunger tracks the motion of the physical plunger (it's a great sign if so), and check if you can launch the ball.

Other physics adjustments

If you can make the plunger work at all with the changes we've covered so far, you might still need to make some further adjustments to make it work *well*. In particular, if the launch speed is too slow or too fast (for example, the ball doesn't make it all the way up the lane even when you pull the plunger all the way back and release), you can adjust that by changing the plunger properties. Here's how:

- Select the plunger object
- Bring up the Properties window
- Adjust the Mech Strength property: *increase* the number if the launch is too slow, *decrease* it if the launch is too strong

Removing a double image

Some tables use their own objects to simulate the visuals of the plunger, hiding the real plunger by positioning it somewhere out of sight. Moving it back into its proper position for the sake of the physics will also make it appear at the proper position visually, so you might now have *two* plungers showing up on-screen.

The easiest way to fix this is to make the built-in plunger object invisible:

- Select the plunger object
- Bring up the Properties window
- Un-check the Visible box under the State section

That leaves the built-in plunger working as far as the physics go, but hides its

visuals, so you get the custom graphics that the table author designed.

Personally, though, I prefer to do the opposite: I prefer to switch tables to use the built-in visuals instead. This usually looks much better than the old scripted visuals. The "new" built-in plunger in VP looks photo-realistic, plus it's very smoothly animated. The custom scripted graphics in many older tables look okay, and some look great, but almost all of them have really choppy animation with just a few "stops" the plunger can appear at.

Getting rid of the old graphics can be a bit of work, so be prepared.

The first step is to change the built-in plunger to use the new visual style that actually looks nice. Existing plunger objects in older tables are always set up to use one of the old visual styles, which all looked crappy. See "Making the built-in plunger look nice" for the procedure.

The rest is a matter of ripping out the game's custom scripted plunger graphics. The procedure is exactly the same as "Replacing the scripts" below, since you want to get rid of the scripts and use the built-in object instead.

Adding a new plunger

If the table you're working with doesn't have a built-in plunger object at all, you've got a bigger job ahead of you.

The first step, at least, is easy. You add a new plunger and move it into position. Adding the object is just a matter of clicking the "Plunger" button in the left pane in the VP editor, then clicking on the playfield where you want to position it.

Make sure that the following options are set in the Properties for the new plunger object:

- Enable mechanical plunger
- Visible

Repeat the steps above under "Try moving the plunger object" to get it into the proper position.

Making the built-in plunger look nice

If you had to create a new plunger object, or you're working with a pre-existing plunger object, chances are that it's using one of the old visual styles that look cartoonish or bad.

The trick to making the built-in plunger look nice is to switch it to the "Custom" style. It's a fairly long procedure, but it's all very straightforward:

- Create a **new** VP table
- On the menu, select Table > Image Manager
- Find CustomWhiteTip in the image list and select it
- Click the Export button
- Save the file somewhere on your local hard disk
- Close the Image Manager dialog and discard the new table
- Go back to the table we were working on
- On the menu, select Table > Image Manager
- Click the Import button

- Find and select the CustomWhiteTip.png file that you saved from the new table above
- Close the Image Manager dialog
- Select the plunger object
- Bring up the Properties window
- In the Color & Formatting section, select PlungerTypeCustom in the Type drop-list
- In the Image drop list, select CustomWhiteTip

Note that there's also a CustomBlackTip object you can use if you prefer a plunger with a black rubber tip - this fits some tables better. You can also create your own original texture if you're familiar with how 3D texture mapping works. That's a bit of an arcane process, though, which is beyond the scope of this chapter.

Note also that there are a bunch of other properties in the Color & Formatting section that let you further customize the drawing, such as setting the diameter of the plunger rod and how many loops of the spring are shown. You can tweak those to get the visuals just right for the table.

Removing the old plunger scripts

Now to the hard part: getting rid of the table's custom plunger scripts.

This is only necessary if you weren't able to get the plunger working by enabling (or creating) and properly positioning a built-in VP plunger object, as outlined above. It's also necessary if you already got the plunger working in terms of the physics but you want to get rid of the old scripted graphics.

Delete fake plunger objects

Start by looking for objects on the playfield around the plunger area that serve no apparent purpose. These will often be one of the following types:

- Wall
- Ramp
- Light

The most likely thing will be a big rectangular object (of one of the types above) covering roughly the same area where the plunger appears. These will usually be somewhat larger than the plunger.

If you find such an object, check its property list to see if it has an associated image in the Color & Formatting section. If so, check what the image looks like in the Image Manager:

- On the menu, select Table & Image Manager
- Find the named image in the list
- Click it and look at the thumbnail

If the image looks like a plunger photo, that clinches it - you've found exactly what we're looking for.

If there's no image, though, it might still be an object of interest. The scripts might be assigning an image in Visual Basic land instead of using the object properties directly. If the object is in the right area, you can make a guess that it's what we're looking for.

If you don't find anything on the playfield, you might try looking at the Backdrop. This is an extremely hacky hack that some table authors use to force an object to appear on top of the table - the "Backdrop" is a sort of super-layer that gets drawn on top of everything (the opposite of what you'd guess from its name!). You might find something called an EM Reel object there:

- Click Backdrop in the left pane
- Note that the Backdrop view is rotated 90 degrees from the regular view, so the plunger area is at the top right in this view
- Look for one or more rectangular objects of type EM Reel around the plunger area

If you find something matching the description - a ramp, wall, light, or EM Reel in the right area and possible with the right image type, try deleting it. Just select it with the mouse and press the Delete key on the keyboard.

Note that might have to "unlock" the object before you can delete it. If it lights up with a gray rectangle instead of blue, it's locked. Right-click it and un-check "Locked" from the context menu.

One good sign that you found the right object is that *there's another object right behind it* that looks exactly the same. A lot of table scripts use a big stack of objects piled one on top of the other as a series of flip-book animation frames. The Visual Basic scripts will make one of the objects visible at a time to animate the plunger motion. If you find a stack of 10 or 20 identical objects in the same spot, you've found an animation flip book. Just keep selecting and deleting them until they're all gone.

Delete scripts

The next (and fortunately last) step is hard to express as a recipe. You're going to be somewhat on your own here. What you have to do now is find and delete scripting code that refers to those objects we deleted above.

If you're good at reading Visual Basic code, you can go through the script and look for subroutines related to the plunger. When you find one, you can usually just delete it outright, from the line that says "Sub *name*" to the matching "End Sub".

A lot of tables with plunger scripts are actually pretty helpful about grouping all of the plunger scripts together in one place. You might find a section marked with comment like this:

```
' *****  
' Plunger scripts  
'
```

If you find something like that, you can try selecting everything that follows that looks plunger-related (based on the names of the subroutines, for example) and deleting it.

If you don't want to pore over the script, there's a pretty easy trial-and-error way of finding the trouble spots. Specifically, you can rely on the fact that all of the plunger scripts will probably make reference to the playfield objects we just deleted. Whenever any of those references are encountered at run-time, Visual Basic will halt the program and throw up an error message telling you that the code is referring to a non-existent object. VB is usually pretty good about showing you exactly where each error occurs, by highlighting the error line in the editor, so this can be a quick and reliable way to find each error and remove the code.

- Run the game
- Wait for Visual Basic to throw up an error message
- If you don't get any errors immediately, try pressing and holding the Enter key on the keyboard to pull back the plunger. Let it pull all the way back, then release the Enter key to release the plunger. If there are any scripts that rely on deleted objects, one of them should trigger an error at some point during this process.
- If you don't get any errors with the keyboard, try again with your plunger device. Hook up the device, move the plunger around, and see if any errors occur at any point.
- Find the line of code where the error occurred
- Delete the code:
 - If the error location is in a Sub..End Sub section that looks totally plunger-related, delete the entire Sub..End Sub section
 - If the error location isn't in a Sub at all, it's probably "initialization" code that runs at table startup. Just delete *that one line* that contains the error
 - If the code is inside a Sub called something like Table_Init, it's also initialization code. Just delete the one line containing the error.
 - If the code is inside a Sub called something like Table_KeyDown or Table_KeyUp, it's keyboard handler code. Don't delete the whole routine; just delete the line that's causing the error.
- Go back and repeat this whole process until the game runs reliably

In most cases, tables with plunger scripting use subroutines that are fairly self-contained and dedicated to the plunger operations. So when you find an error, you can usually just delete the entire subroutine that contains the error line: that is, all lines from the nearest **Sub name** that precedes the error line to the nearest **End Sub** that follows it.

Something that's likely to happen during this process is that you might delete one subroutine block that *another* subroutine calls. So after you delete the code with the first error, run again, and you'll probably hit a *new* error in code that's trying to call the code you just deleted.

Getting the table working now is usually just a matter of repeating this seek-and-destroy mission until until you stop getting errors.

If all else fails

In some cases, the "delete code until it starts working" approach can backfire. Some tables are just too complicated, with Visual Basic code that's too intertwined, for mere code deletion to work. You sometimes have to rework the code a bit instead.

If you know VB well enough, you can analyze the code and figure out what you have to change. If you're not a VB power user, well, it's a bit beyond scope of this section to turn you into one. My advice would be to ask nicely for help on the forums - maybe you can find the original table authors and persuade them to update the table to modern standards, or enlist help from one of the many Visual Pinball power users on the forums.

40. Coin Door

If you want your pin cab to look authentic, a standard coin door is a must. The coin door has been a fixture on pinballs since the 1950s, and the design has been fairly uniform since the 1980s. It's so ubiquitous that I'd consider it part of what makes a pinball machine look like a pinball machine. Even if you haven't taken conscious notice of the coin doors on real machines up until now, you've probably seen them often enough that you'd notice that something was missing if you didn't include one on your virtual cab.



Beyond the cosmetic value, the coin door also has some practical, functional features that make it worth including. If you've only experienced pinball as a player, as opposed to as an owner or operator, you probably only see the coin door as the place you feed in quarters. But for a machine you own and operate, which your cab will be, the coin door actually has some other important functions.

For one thing, the coin door is the access port for opening the machine up for maintenance work. Opening the coin door lets you access the lever that releases the lock bar, which in turn lets you take off the top glass, which in turn lets access the playfield, and (on a real machine) lift up the playfield to get access to its underside and to the interior of the cabinet.

The coin door is also the place where you'll find the "service buttons": a cluster of three or four push-buttons inside the coin door, which are let you access the operator menus in ROM-based tables. These menus are where you set the game options: Free Play mode, 3 ball or 5 ball mode, replays as extra balls, and so on. In a virtual cab, you can always use your PC keyboard to access the menus, so the physical menu buttons inside the coin door aren't absolutely required, but I think they're a lot more convenient.

The inside of the coin door is also an excellent place to conceal any added custom controls of your own. Some people put the volume knob for their audio amplifier here, for example. It's obviously not a great place for anything you'll access frequently, but it's ideal for any operator controls that you only need to access once in a while for setup purposes.

Finally, the coin door is useful for its nominal purpose as the place to insert coins. Even for a virtual cab, it's useful to be able to feed in quarters - either for real or in a virtual way, such as by pushing a button. In either case, the coin door is the ideal place for the coin chutes and/or coin buttons. The reason it's important to have some kind of coin feature in a virtual cab is that some tables don't have a Free Play mode, so the only convenient way to play them is to make them think you're adding quarters. The early solid-state ROM-based games in particular tend to have very primitive software with few or no configurable options. A lot of the older EM (electro-mechanical) tables also lack coinless play options, although for those you can at least add it yourself via scripting. In any case, it's nice to have an easy way to add coins to deal with tables you can't set to Free Play. One way is to make the physical coin chutes functional, and another is to add a button that simulates inserting a coin. I personally like having both options. Both approaches are covered later in this chapter.

Buying a coin door

If you're salvaging an old pinball machine to use as your cab's body, and it already has a coin door installed, you're set. If you're building a new cab from scratch, you can buy a new pinball coin door from an arcade parts supplier (see Chapter 4, Resources). You can also look for used ones on eBay, although as always, check prices for new parts first: used pinball parts on eBay tend for whatever perverse reasons to be more expensive than new ones.

One really nice feature when you're buying a coin door is that most of the different types can be used interchangeably, since they're all designed to fit the same cutout. They've been using a standard size since about the mid 1980s. Doors designed for Data East, Stern, Williams, and Bally machines will all fit the same the standard cutout, as long as they're for machines from the 1980s or later. However, the different makes do have some differences on the inside, such as the type of service buttons and the type of wiring plugs used.

Here are the top options currently available commercially:

- Williams/Bally WPC-era coin door, Williams part numbers 09-17002-26, 09-23002-1, 09-37001, 09-46000, 09-61000-1, 09-61000-1, and 90-96017. These typically come fully assembled, with the lock and key, slam tilt switch, brackets for two coin mechanisms, coin chute lamps, coin switches, and service buttons. All of the electronics are pre-wired to a 13-pin connector. The only parts not included are the coin mechs (the little Rube Goldberg devices that check inserted coins to make sure they're real). You can buy those for about \$10 apiece and pop them into the chutes if you want to be able to use real coins.

This is the door I used on my cab. I like it because it exactly replicates the look of the Williams 1990s machines, which is the cohort that includes most of my all-time favorite games. It does have a couple of downsides, though: it's more expensive than the SuzoHapp door below, and the design makes it way more difficult than it should be to replace the coin inserts (the little lighted coin chute labels saying "insert quarters").

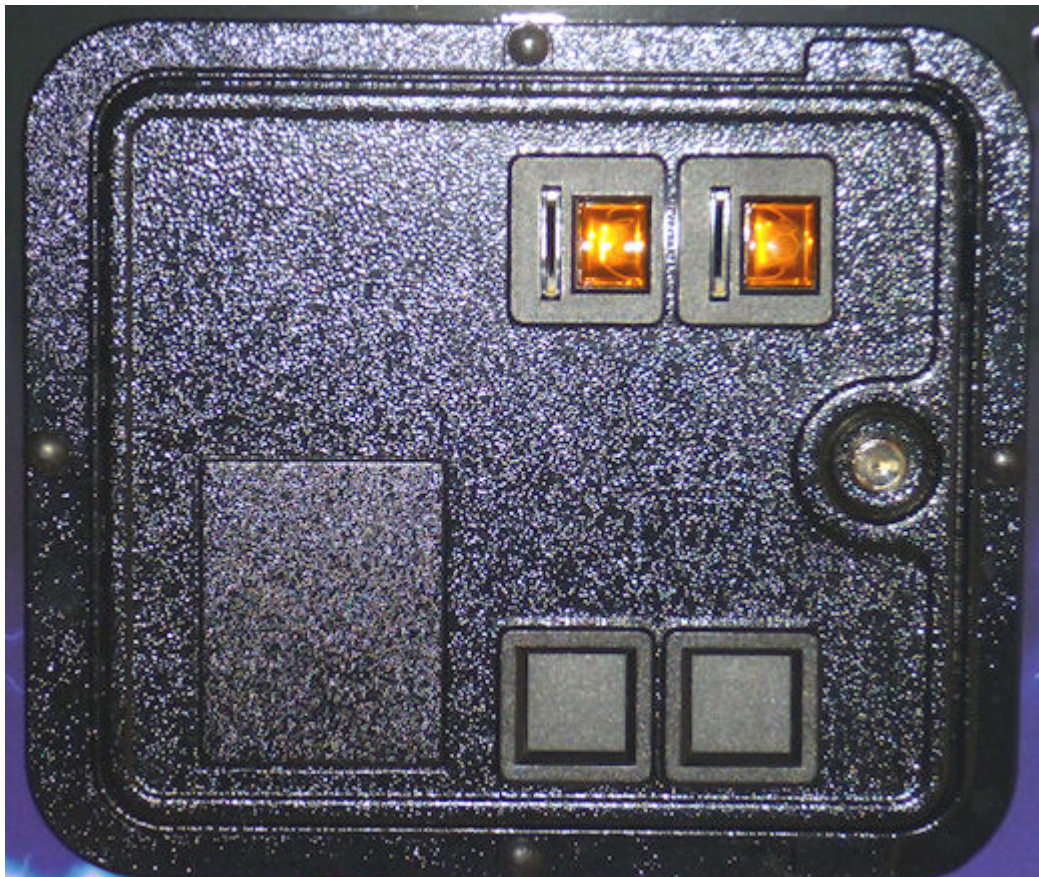


- SuzoHapp universal coin door, manufacturer part number 40-0696-30. This is a third-party replacement part compatible with OEM parts for almost any machine from the mid 1980s or later. It's also the factory-installed part on some of the newer Stern and Jersey Jack machines. This one comes with all of the coin slot hardware installed, including the coin mechs (for quarters), but it doesn't have any pre-installed wiring and doesn't include service buttons.

This is my top pick for the DIYer. It's about 40% cheaper than the pre-assembled Williams door above, and it's functionally equivalent after you add the service buttons. The lack of pre-installed wiring might actually be a plus in a pin cab, since you can wire it with your own connectors instead of the obscure Molex connector used on the WPC doors. The design is also nicer in some ways than that of the WPC doors: it's easier to install the coin mechs in this door, and it's *much* easier to install custom coin slot inserts. The only downside, really, is that it doesn't look exactly like the original WPC doors, although it's close enough that most people wouldn't know the difference.

If you want service buttons (which I think you do!), you can easily add them as an extra part. Buy the **Stern 4-button service assembly** (part number 515-1963-00). Note that there's a similar service button panel designed for the WPC doors, but it has a different mounting bracket that won't fit this door, so be sure to buy the Stern version. You get the same set of buttons with either assembly, so there's no functional difference; it's just the mounting hardware that's different.

One other minor feature missing in this door vs. the WPC door is a slam tilt switch. You can add that separately if you want 100% parity with the Williams door. I personally don't see any functional reason to include a slam tilt switch on a virtual cab, since it's only present on the real machines to discourage extreme abuse that you'd never subject your home machine to. But you can certainly include it for the sake of completeness if you like.



- Stern coin doors, for SAM machines (Stern part 501-5018-172) or SPIKE machines (501-5018-173). SAM and SPIKE refer to the last two generations of the Stern platform, not to specific table titles. These are functionally about the same as the Williams door listed above. The main difference from the Williams door is that these use different connector plugs for the electronics. The SPIKE version is actually the same equipment as the SuzoHapp option above, but adds pre-installed service buttons and wiring.
- Data East coin door. These come bare-bones, with nothing installed except for the key and the basic coin slot brackets (no coin mechs). As a result, they're cheap. They're a good budget option if you only want the facade of a door (without all the functional bits) for cosmetics. I wouldn't recommend this if you *do* want all of the functional elements, though: you probably won't save any money after adding all of the parts it's missing compared to the turn-key options above, and it'll be a lot of hassle to source the parts and assemble everything.
- Smaller video game coin doors. If you're building a mini-cab, the regular pinball coin door might be too large. There's a narrower type of door commonly used in video games. Try SuzoHapp for a variety of options.
- Fakes/Decals. If you only want the cosmetic effect of a coin door without any functionality, you could use a 3D-printed plastic façade, or simply a custom-printed decal with a photo of a coin door (see Chapter 22, Cabinet Art for more on decals). This might let you squeeze out a little cost if you're on a tight budget, or if your cab is too small for a real coin door.

Coin mechs

Coin "mechs" (mechanisms) are the gadgets that validate coins inserted through the slots in the door. These use a modular design, with a standardized physical form factor, that lets you swap in mechs for different types of coins or tokens. If you want to use quarters, you install a mech that takes quarters; if you want to accept arcade tokens, you install a token mech.

Your new coin door assembly probably came without any mechs installed - just empty brackets where the mechs go. So if you want your pin cab to accept quarters or other coins, you'll have to buy the mechs separately and install them in your coin door.

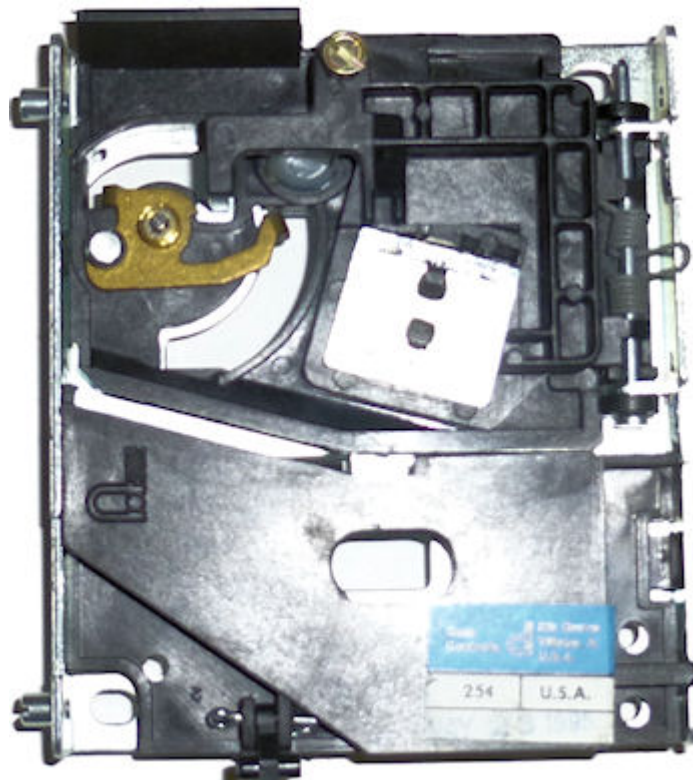
US coin doors (anything from the 1990s to present) are almost always set up with two coin slots fitted with brackets for the standard coin mechanisms. The pinball vendors (Pinball Life, Marco Specialties) sell mechs that accept US quarters. Those run about \$10 each.

You can also find mechs that accept other coin types besides US quarters, but not at the pinball vendors; they only sell the quarter mechs. Here are some leads on where to find mechs for other coin types:

- SuzoHapp sells mechs that accept Canadian coins (quarters, loonies, and toonies), plus mechs for several types of arcade tokens. They also sell matching arcade tokens.
- Coin Mechanisms, Inc. sells acceptors for nearly all types of US coins, plus acceptors for numerous other countries' coins. Navigate to **Coin Doors, Coin Mechs & Face Plates > Mechanism Coin Mechanisms**.

- You might also find mechs for various coin types on eBay, but be careful that you're buying a compatible mechanism. I'd consider only **mechanical** coin acceptors, not anything electronic. The electronic mechs are usually meant only for Asian and European markets, where the coin doors have a different setup. The electronic mechs usually won't fit a US coin door, and even if they do, they might not interface easily to a virtual cab's key encoder.

Here are a few pictures to help you identify the right type by physical form factor, if you're looking for a non-quarter denomination on eBay:





Note that your mechs might have the stubby little posts coming out of both sides. The SuzoHapp/Stern doors need all four posts to seat properly, whereas the WPC doors only use the rear posts. The pictures above show the WPC configuration with only the rear posts in place. If your mech has all four posts, and you have a WPC door, no problem: you can easily remove the unwanted extra posts by unscrewing them.

"Any Coin" dummy coin mech

The standard coin mechs described above are designed to accept specific coins or tokens. Since they're sold for commercial use, they're designed to validate the coin's authenticity by size and weight. They'll reject anything besides the correct coin type.

For home use, you might not care about the validation part, since you might want to use tokens or assorted coins instead of keeping a supply of quarters on hand. In that case, there's an alternative dummy coin mech available that accepts just about anything resembling a coin, without trying to validate it:

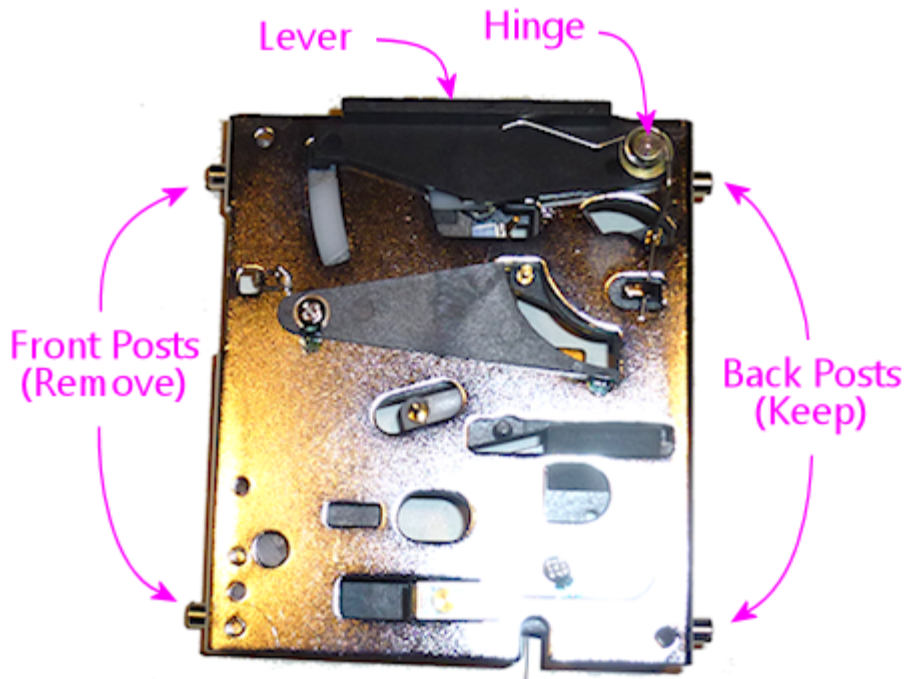
ArcadePlastics.com "Any Coin" replacement mechs

The Any Coin mechs are drop-in replacements for the standard mechs, so you should be able to use them in any of the common coin doors. The price is about the same as the standard mechs, so the main reason to buy them would be that you like the option of using assorted coins and tokens instead of just quarters. They might also be less likely to jam than the standard mechs, since they don't need the intricate internal maze that the standard mechs use to reject invalid coins (although I don't find that the real mechs jam much either).

I'm personally happier with the standard mechs, just for the sake of authenticity. But then again, I rarely use coins anyway when playing games on the machine, so it makes little difference at a practical level.

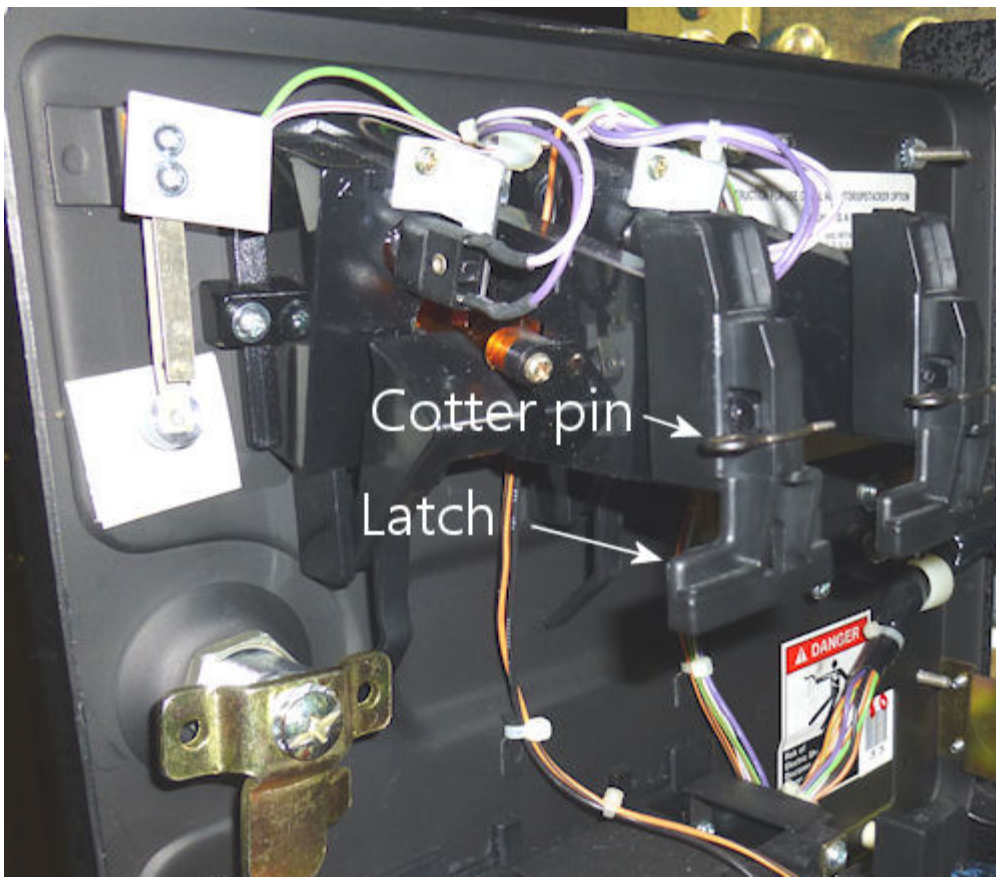
How to install a mech in a WPC door

If your coin mech has **four** of the little stubby posts, remove the ones on the front side - the side that faces the coin door. Simply unscrew the posts and set them aside (you can put them in your misc parts drawer, or just discard them). The mechs often come with all four posts installed, because some other coin doors require them, but they won't fit into the WPC-style doors.



Coin mech with all four posts installed. You'll have to remove the posts on the front (door) side before installing the mech in a WPC-style door.

Open your coin door and find the brackets that hold the mechs. These are right behind the coin slots. You should see a couple of big latch levers near the top of the door, as pictured below.



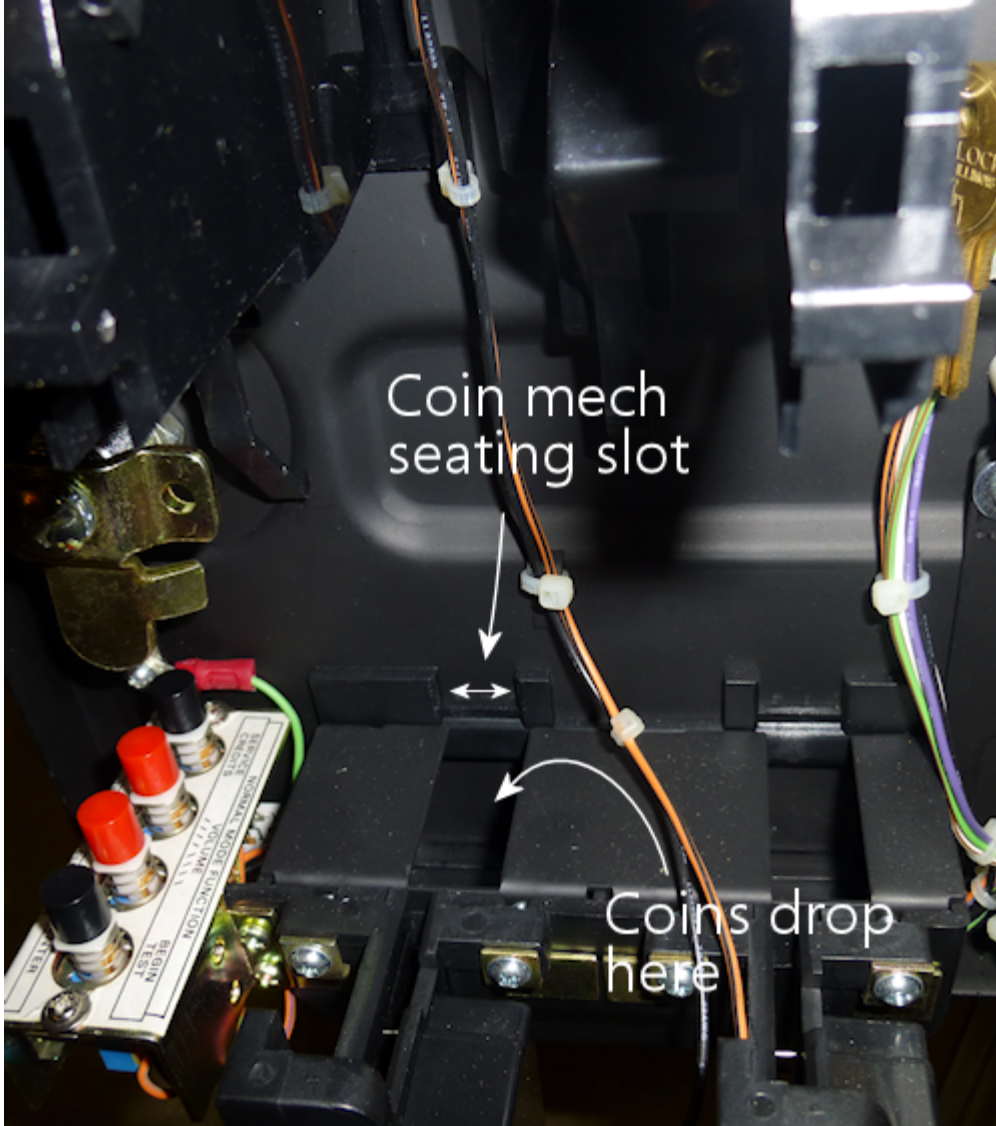


There might be a cotter pin installed in each latch to lock it in place; if there is, remove it. Pull the latch outward from the bottom and flip it all the way up. (It might offer a little resistance when all the way down, but you shouldn't have to exert too much force.)

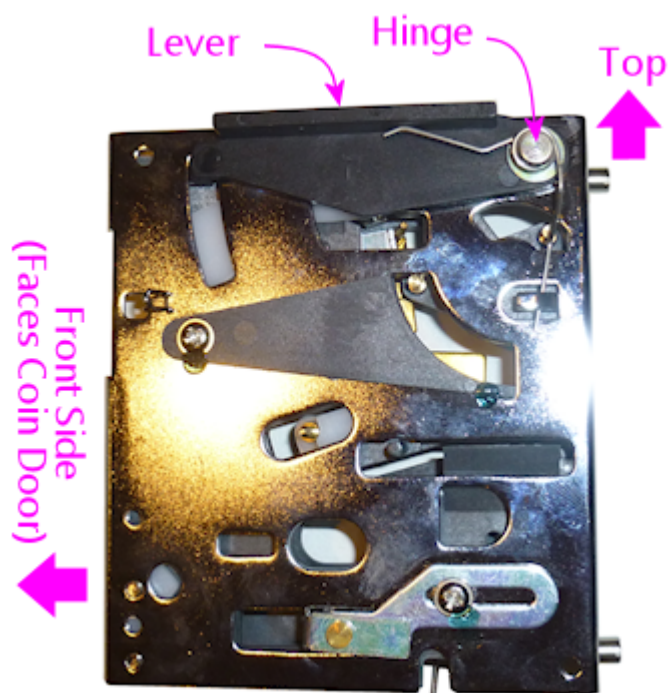


The coin mechs fit in the space below the latch, in the area marked in the photos (above and below) as the "seating slot". This area isn't very well delineated - figuring out exactly where the mechs should land is the hardest part of installing them, in my opinion. The picture below shows more of a head-on view that might help. The "slot", such as it is, is that open space between the two stubby little protrusions along the front wall. You want to fit the front bottom corner of the mech between those nubs.

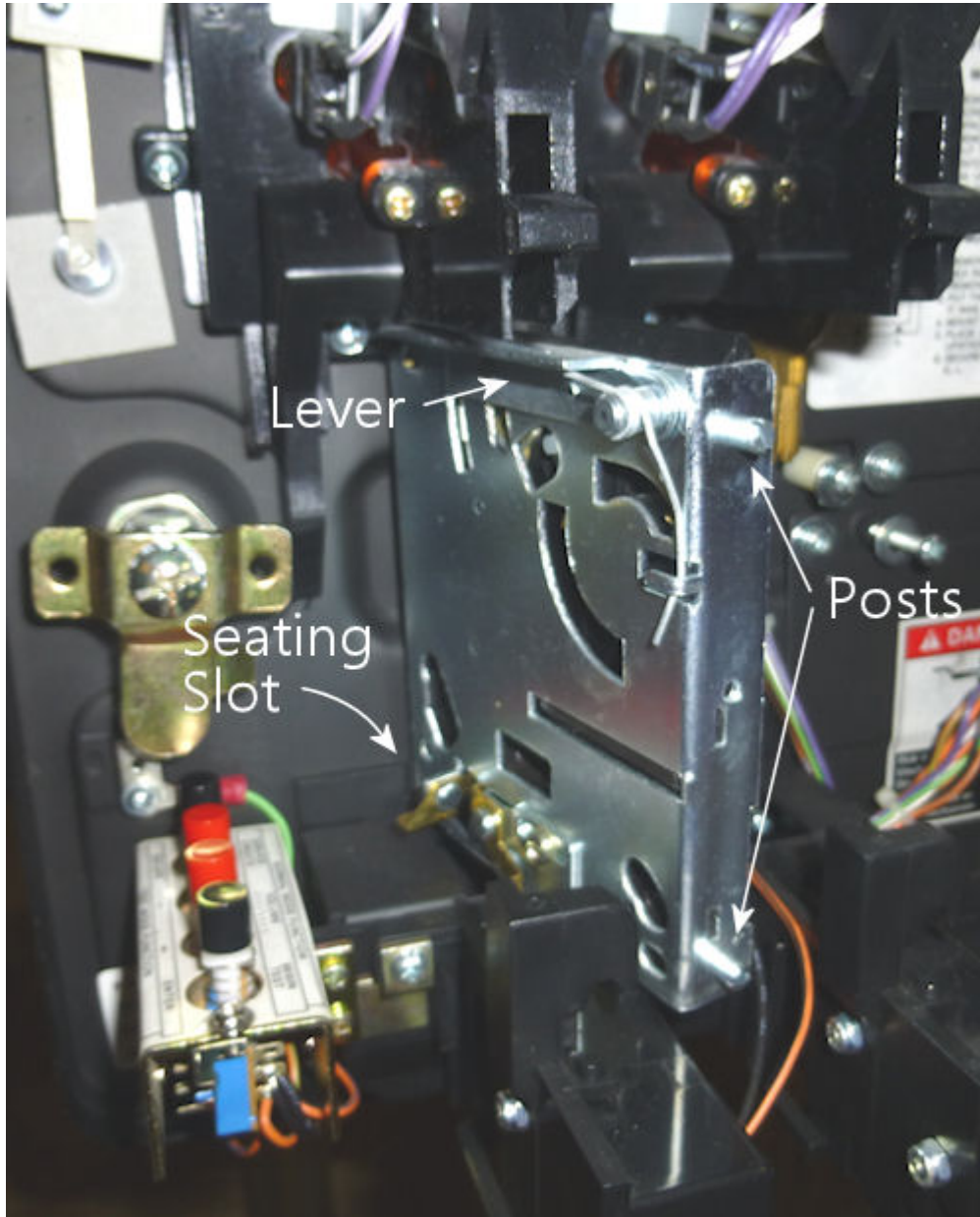
That big adjoining opening in the floor is **not** part of the seating slot, by the way. It's the opening that the coins drop through. Don't try to squeeze the mech in there; it'll end up resting right on top of that opening.



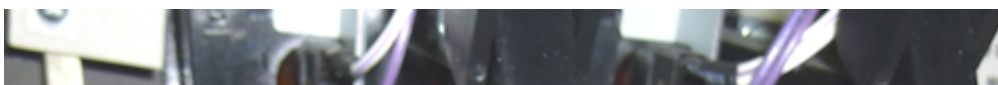
To orient the mech, find the big plastic lever that's alongside one of the edges. (The lever is engaged when you press the "push to reject" button to clear a coin jam.) When you install the coin mech, install it with the lever facing up, and the hinge towards the back, facing away from the coin door. The hinge side should also be the side with the two posts.

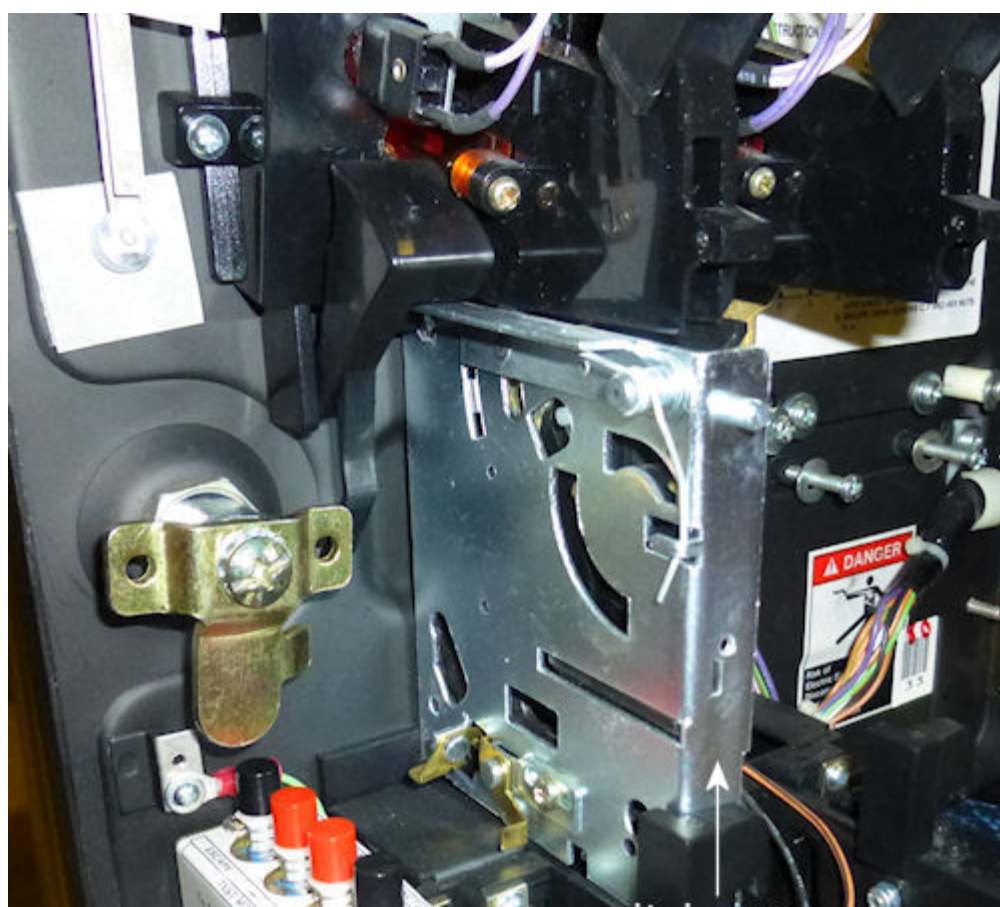
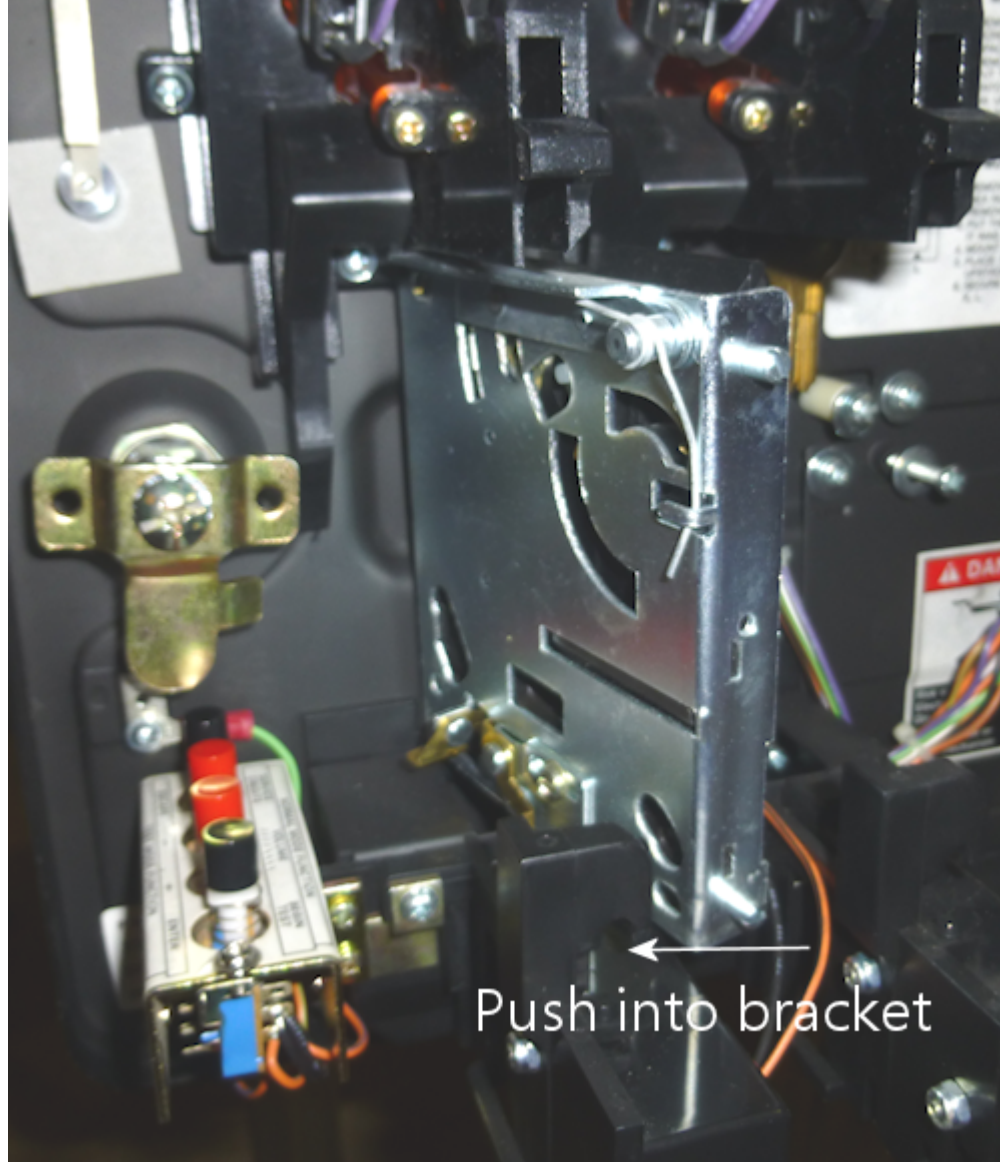


Insert the coin mech into the slot. You'll have to insert it at an angle, as shown below, to get around the back end of the bottom bracket. Orient it as shown, with the little metal posts sticking out the back, and the metallic side of the unit facing you. At this stage, there won't be any sensation that you're fitting the mech into a slot; it just kind of sits there. But the bottom front corner should be nestled between those two little nubs we pointed out above, and you should push the mech as far forward as you can against the door.



If everything's aligned correctly, you should now be able to push the bottom rear of the mech - the part where the bottom post is sticking out in back - into the bottom bracket. This should straighten things out so that the mech is square with the door. This will be a fairly tight fit, but it shouldn't take a lot of force. If it doesn't slide into the bracket fairly easily, the mech is probably sticking out too far in back because you don't have the front aligned correctly with the nubs. Try wiggling the front to get it more completely into the little recess between the nubs. You might also have to slide it upwards slightly to fully seat.

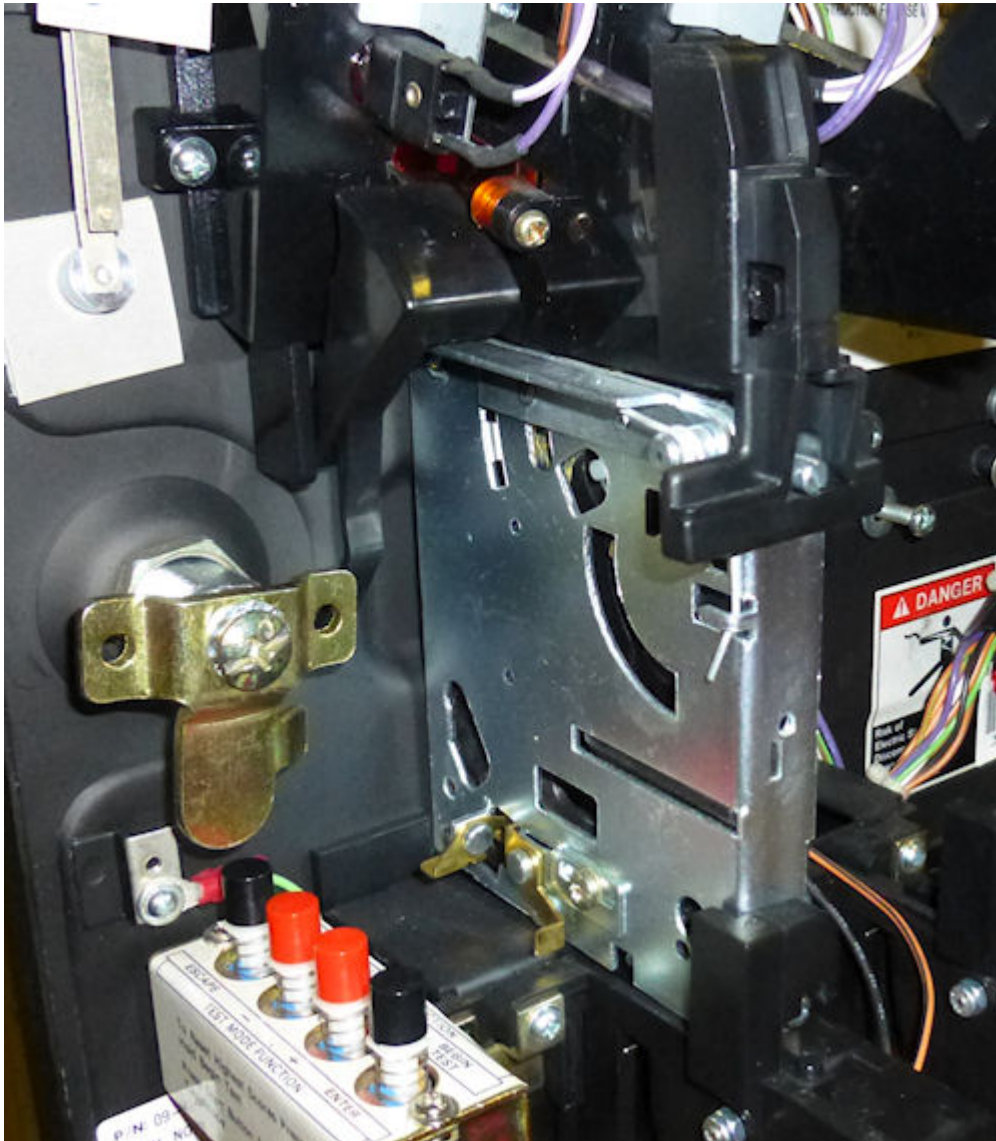






Now flip down the top latch (the one we flipped up in the first step). It should fit over the top post at the back of the mech. If it doesn't seat, try sliding the mech upwards a little, and try moving the top back and forth a little to fit into the lever.

If you took out a cotter pin at the start of the procedure, re-install it now.

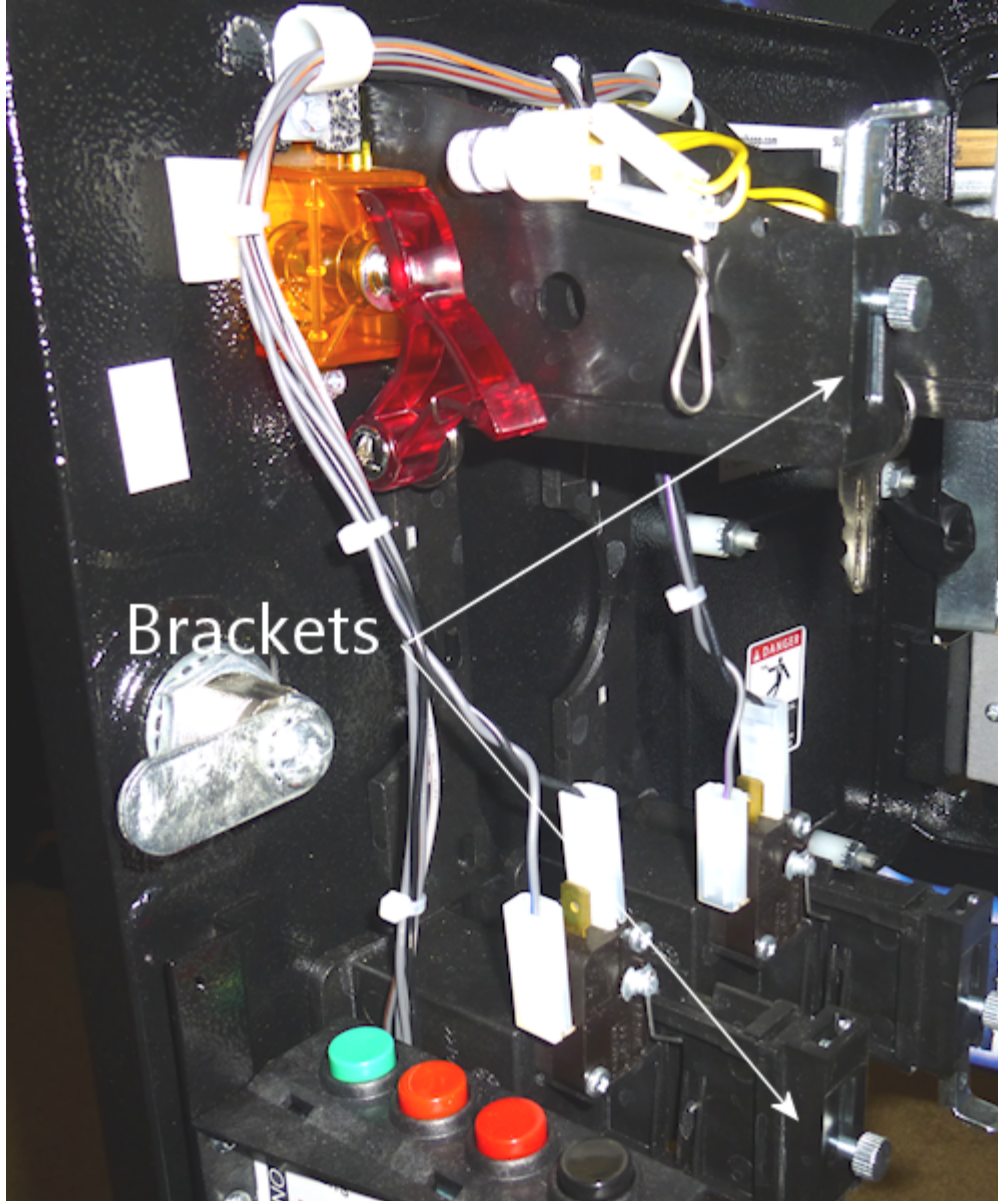


Done! If you have a second coin mech for a second slot, the procedure is exactly the same for that.

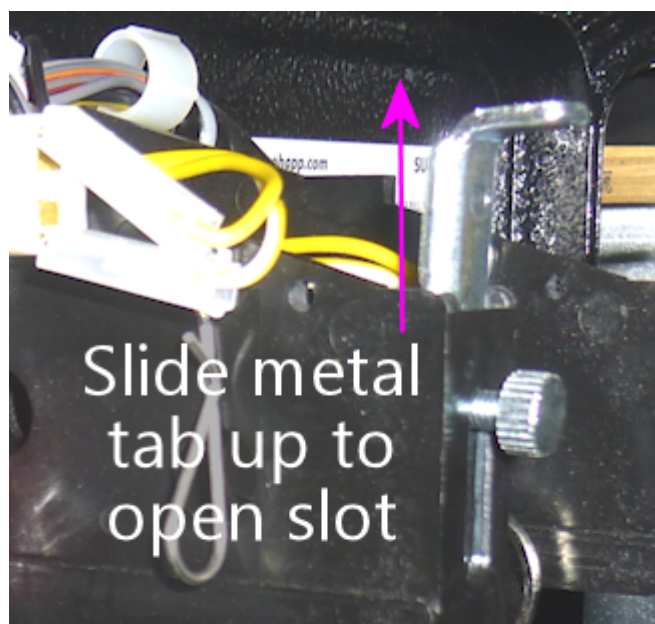
The blank space in the left half of the WPC-style door is designed to accommodate a paper dollar bill acceptor. I haven't installed one of those myself, so I don't have any installation instructions to offer (and I doubt that most virtual cab builders would be interested anyway). Some cab builders use the space for other purposes, such as audio volume knobs or extra buttons.

How to install a coin mech in a SuzoHapp/Stern door

Open the coin door and find the brackets that hold the mechs. They're right behind the coin slots.

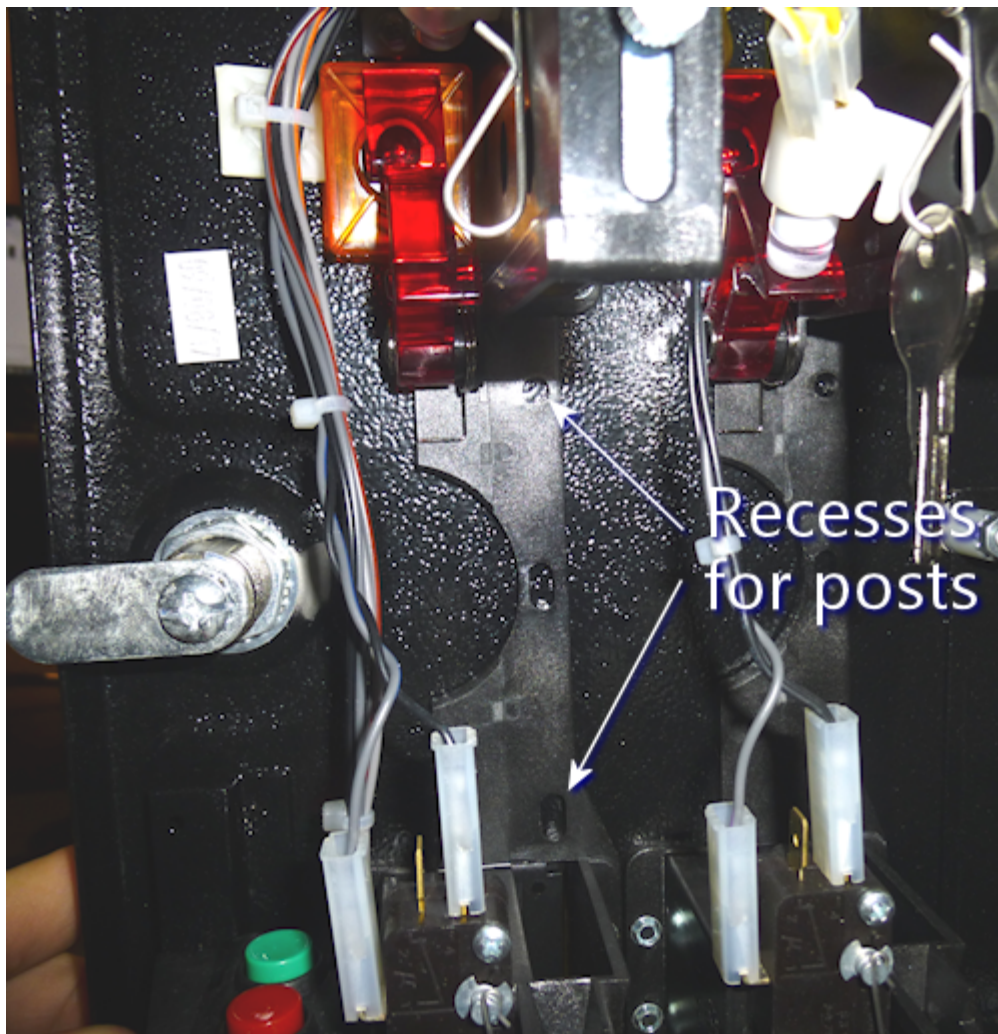


Make sure the metal tabs at the top and bottom of the slot are in the "open" position, meaning they're out of the way of the slot. Slide the upper tab all the way up, and slide the lower tab all the way down. Loosen the set-screws if necessary. (The brackets might take a little effort to slide up and down even after loosening the screws, but of course don't force anything.)



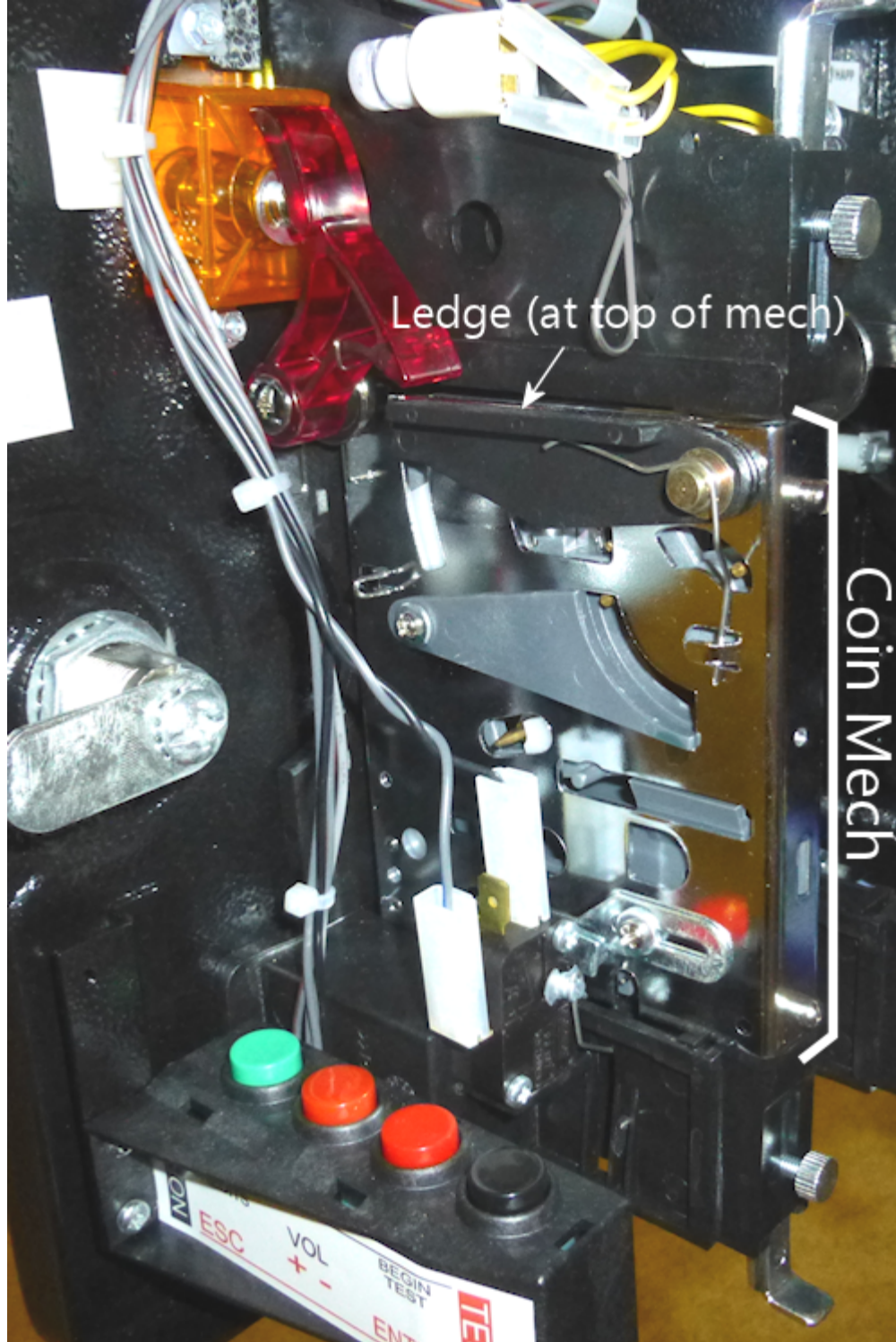


At the front of the area where the mech will sit, you should see two little circular recesses at the front of the slot. These are where the posts on the front edge of the coin mech fit.



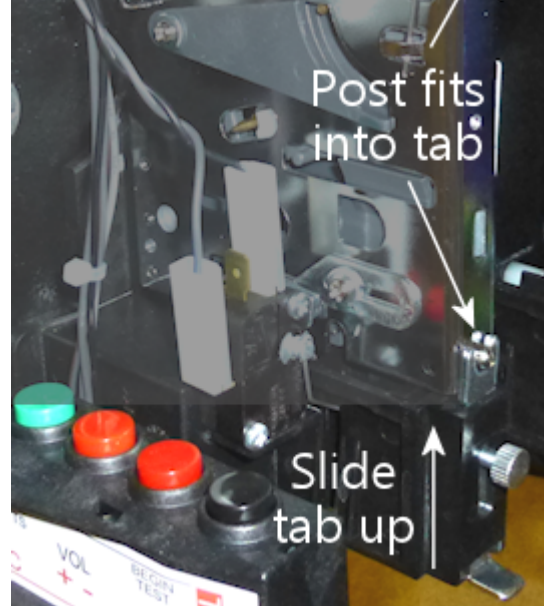
Slip the mech into the space, fitting the posts on the front edge into the recesses. Orient the mech as shown below, with the metallic side facing you. You can also identify the orientation by the "ledge" that sticks out on from one side of the mech. The ledge is at the top, and should be on the side facing you.





The back of the mech should now align with the metal brackets above and below. Slide the brackets so that the posts on the back of the mech fit into the openings on the brackets.





Now all that remains is to tighten the set-screws to fix the mech in place. Done!

European coin doors and coin mechs

My understanding is that the coin doors sold for European markets use a different design that's not compatible with the mechanical quarter acceptors used in the US. The European coin doors are set up for electronic coin acceptors that can be programmed for multiple denominations, to accommodate a wider range of coins. The electronic coin acceptors have a different physical form factor from the mechanical ones used in the US, so they need different brackets and have a different installation procedure. I'm afraid I don't have any experience with these, so I have to leave it up to you to figure out the procedure. (If anyone wants to supply me with a photo sequence for European doors, I'll be happy to add that here.)

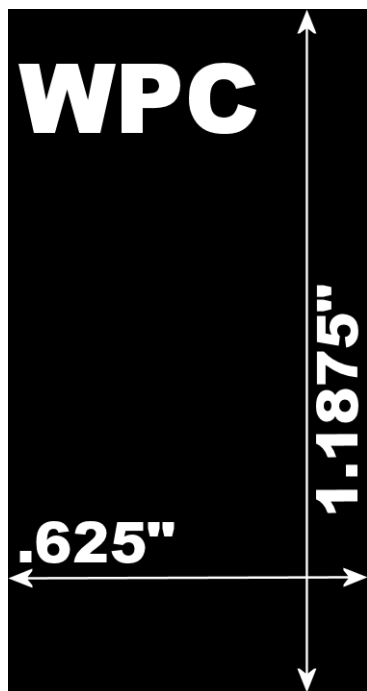
Custom coin slot inserts

Each coin slot on the coin door has a "reject" button at the top for clearing coin jams. These usually come with pre-printed legends indicating the type of coin accepted, usually "Quarters" or "25¢" for a US door.

The legend is printed on a little piece of transparency film sandwiched inside the button, so you can take that out and replace it with a custom label. This lets arcade operators swap in labels saying "Tokens Only" or "SBA Dollars", for example. For a virtual cab, it's an opportunity to personalize your machine.

You can buy pre-printed inserts for common denominations, but it's pretty easy to create your own custom inserts with a laser printer. To help you get started, here are the dimensions for the most common coin door types:





If you're not sure that your inserts match one of these layouts, you can just measure the old one currently in your door after extracting it, which we'll get to in the "how to install" sections below.

Your inserts will look best if you use white text and graphics on a black field. That's the way the "real" ones look. The black background should ideally be completely opaque, which is why you need a laser printer; the ink used in ink-based printers just isn't opaque enough.

When I created my labels, I had good results with plain white paper printed on a laser printer. Your mileage may vary, though; laser printers certainly vary in how darkly they can print. I've seen posts on the Web from other people who tried to create custom inserts this way and weren't satisfied with the opacity. If you try it and find that the results look too washed out when back-lit, you might try a double layer approach: print a bottom layer on white paper, and a top layer on laser printer transparency film. Cut out the two pieces and carefully align them to overlay the graphics. The double layer of toner should greatly improve the opacity of the black, and since the top layer is clear film, it won't (much) dim the white parts. You can try two layers of white paper if you don't want to spring for the special laser transparency sheets, but the white parts will probably be too dim with that approach.

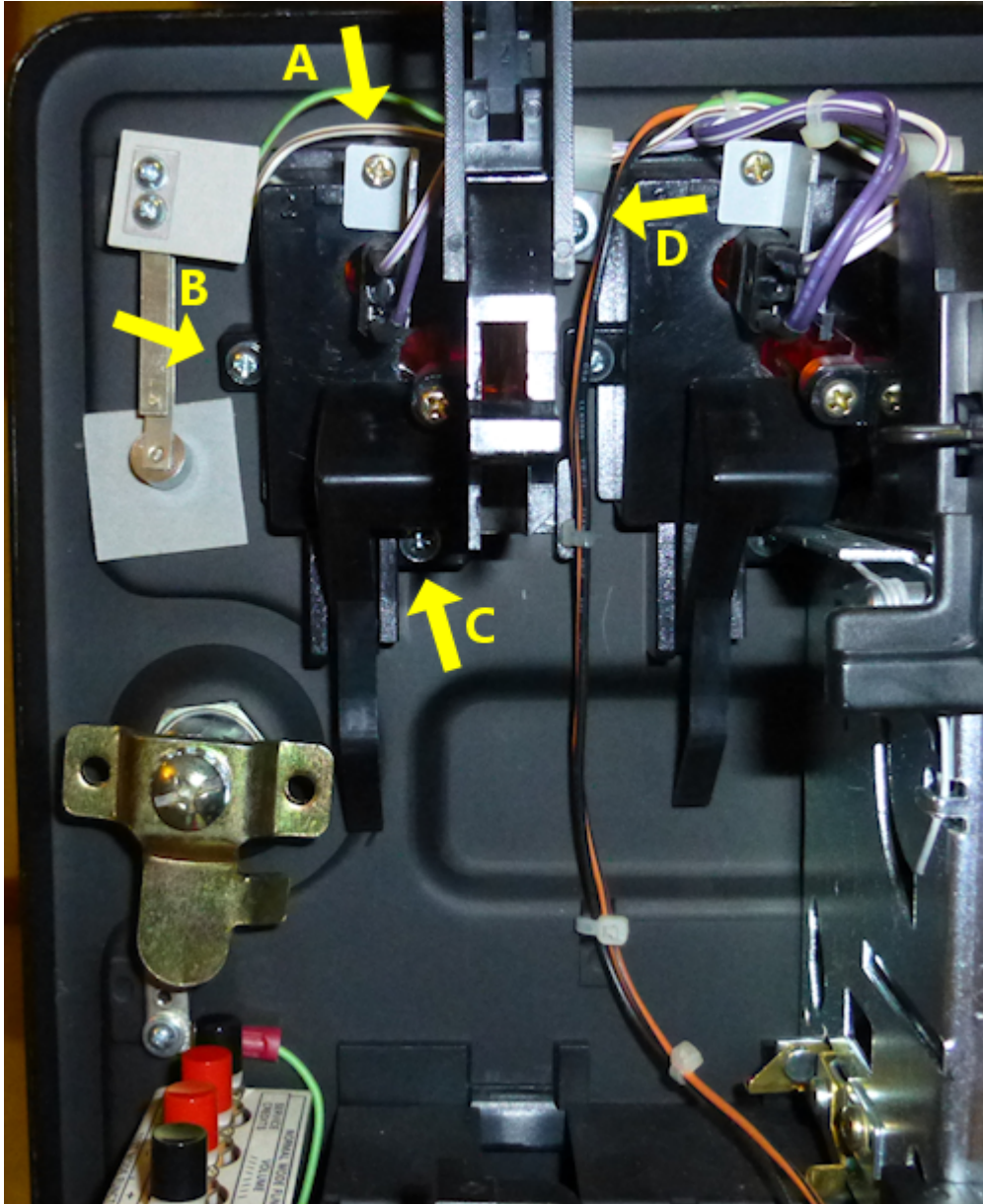
Once you have the inserts printed, you still need to install them in the buttons. The exact procedure for that depends on the type of coin door you have.

I think white paper makes a better base layer than making the whole thing transparent, by the way, because white paper will do a better job of diffusing the back-lighting. The original inserts that come with your coin doors will probably be printed on thin white plastic film, which is probably even better than paper, since it doesn't have the visible grain that paper has. You can find laser-printable translucent film sheets on Amazon and at art-supply shops, but I haven't tried any of them myself. (Try searching for "backlight film" or "lightbox film".) If you've used a particular product that you'd like to recommend, let me know and I'll pass along the recommendation here.

Replacing inserts in WPC coin doors

Fair warning: this process is a bit of a pain with the WPC doors. (Which I hope you won't find to be too much of an understatement.) I'd strongly recommend doing **one chute at a time**: leave one chute fully assembled while working on the other one. That way, if you get stuck during re-assembly, you can use the still-assembled one as a reference to figure out how things are supposed to go. This could be especially important if your door doesn't exactly match the one I used to formulate these instructions. I'm sure there are some little variations in different versions of this product that they've shipped over the years.

To get to the orange button, you have to take off the entire top coin mech bracket. Start by removing the coin mech itself (reverse the procedure for installing a coin mech in this door, described above). Then remove the four screws shown below.



The screw labeled "A" above attaches the bracket for the lamp that back-lights the button. Pull the lamp out of the bracket (it'll be sitting in a hole in the back of the bracket). You **don't** have to disconnect the lamp or disconnect any wires. Just let the lamp dangle from the wire.

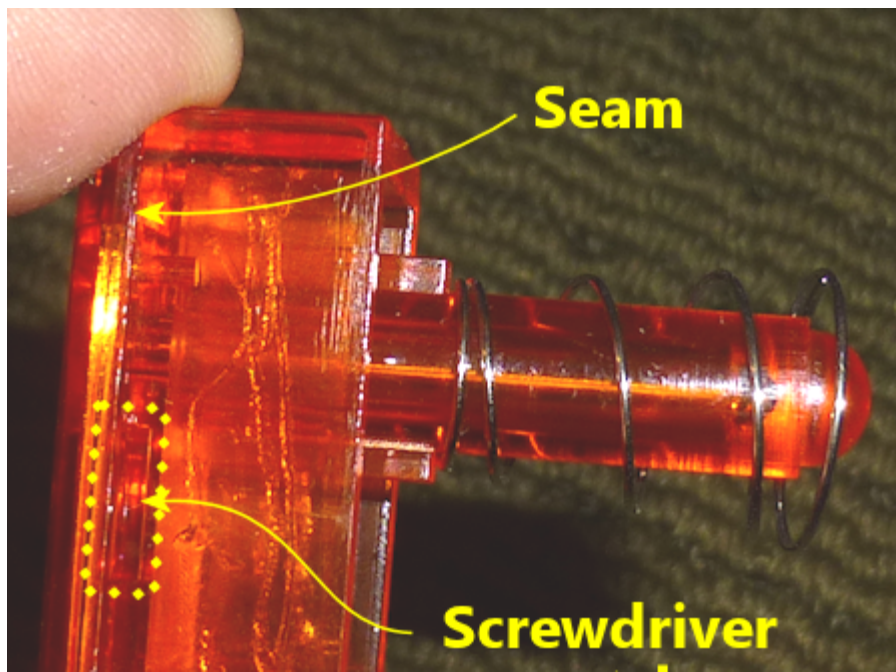
Screw "D" holds down both the bracket and a wiring clip. Try to keep the wiring clip attached to the wire to make it easier to put things back the same way later. (I did warn that this was going to be a pain.)

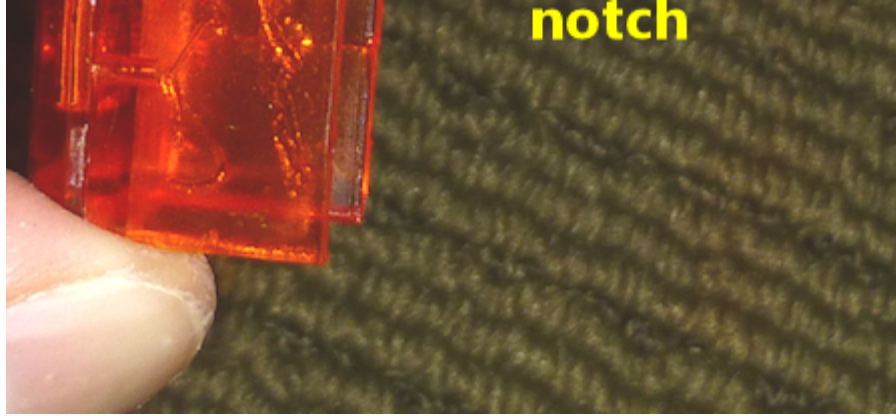
The bracket should now be free, so gently remove it. It might be a little tangled in the wiring, so if it doesn't come right out, gently ease it out from under any wires pinning it down. Again, you shouldn't have to disconnect any wiring at any point in this entire process.



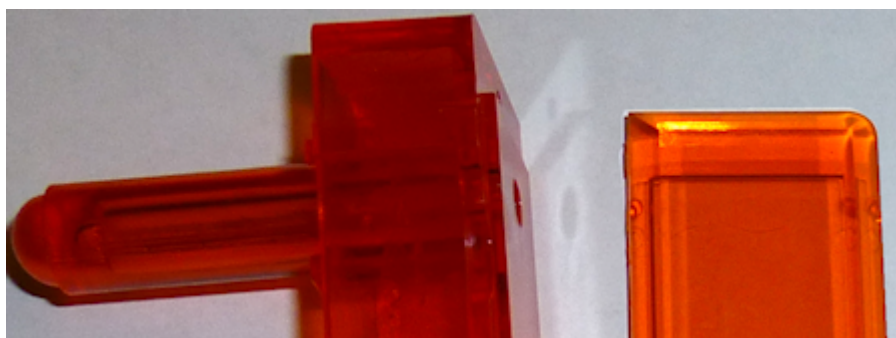
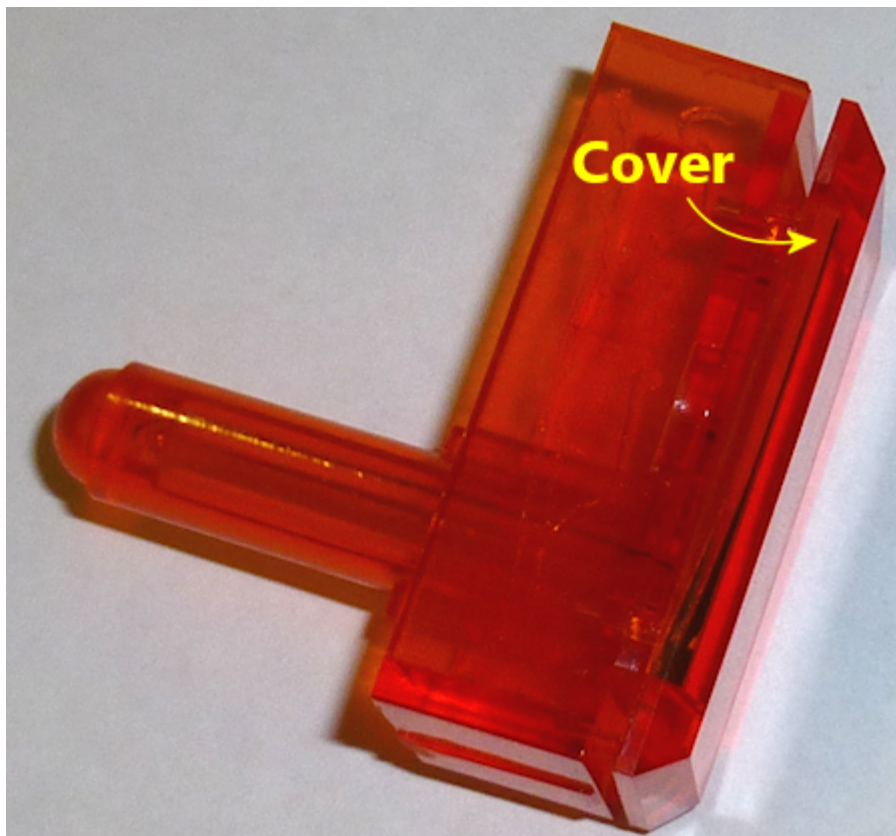
You can now easily remove the orange button. In fact, it'll probably pop out on its own when you pull out the bracket, because there's a spring actively pushing it away from the door, and the bracket was the only thing keeping it in place.

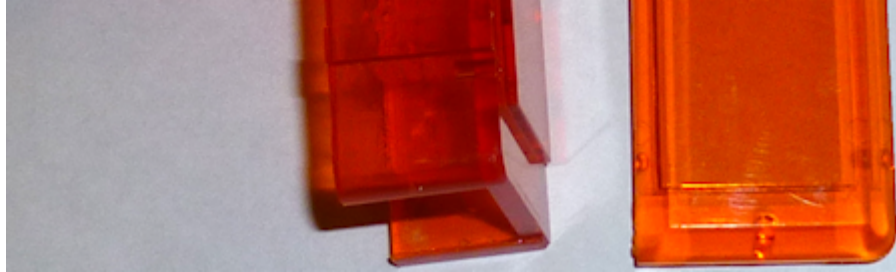
Note that you'll want to remove that entire black bracket that the button is housed in, even if the button came out before you had the bracket all the way out. It'll be easier to get it back together later if you just take the bracket out entirely.





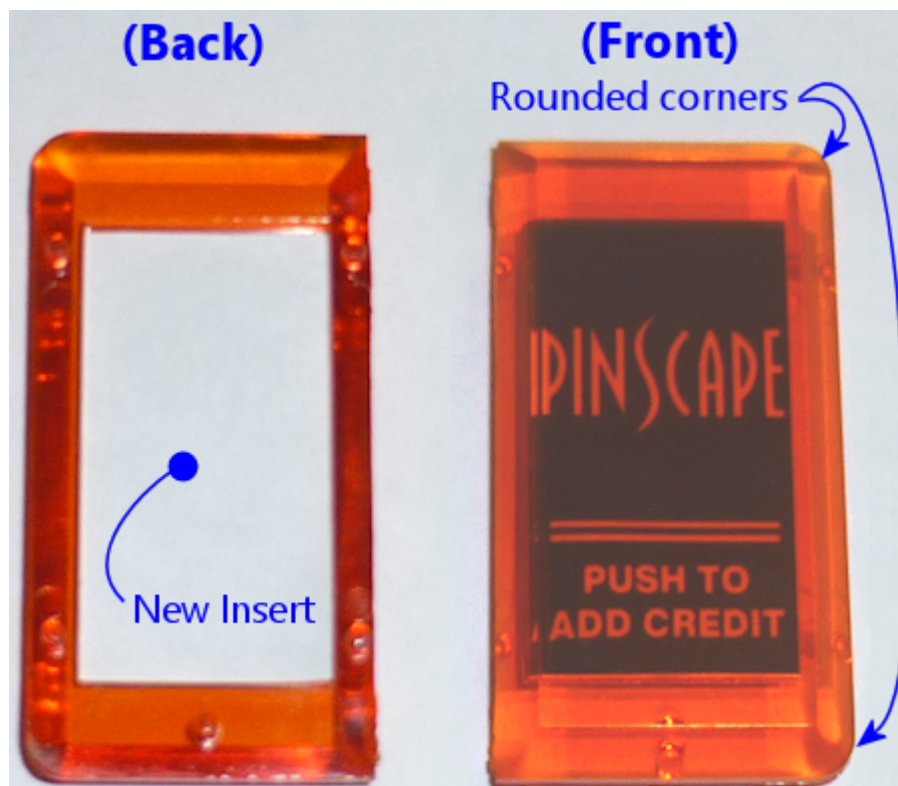
We've almost reached our objective of removing the old insert. The orange button might look like a single piece of molded plastic with the insert forever locked inside, but it's not. There's a removable cover, and the insert is just under that. You should be able to see the seam pretty easily about a millimeter from the front. The cover isn't molded in or glued on; it's just held in place by four friction pins. You should see a notch on each side along the seam. You can stick a flat screwdriver in there and twist gently to pry the cover off. Be gentle so that you don't break the little plastic pins, and do the prying as evenly as you can. Pry a little on one side, then pry a little on the other side, and rock it back and forth like that a few times until the cover comes loose.





Once you have the cover off, just take out the old plastic film and insert the new one. The cover has a very slight recess for the insert, so I'd put the cover face down on a table, put the new insert into the cover, and then reattach the back to the cover, keeping the cover lying flat so that the new insert stays put. Align the pins and press the two pieces together until flush.

Pay attention to the subtle asymmetry of the cover! One side has curved corners, and the other side has square corners. You'll want those to align when it's back together, so make sure you install the new insert right-side-up.



Put the spring on the button stem, noting that the narrow end of the spring faces the button.



The rest is just a matter of reversing the disassembly steps. It's a bit of a hassle to get the bracket back in the proper position in the door, especially since the spring-loaded button wants to keep falling out. Be patient, and you should be able to get it seated again. Remember to look through the coin button opening in the front side of the door as you get the button seated - that'll make it easier to get everything aligned again.

When you have the bracket aligned properly (with the coin slot and button in the right place as viewed from the front side of the door), hold it in place while you screw in the bottom screw (screw "C" in the first photo above). That should be the longest of the four screws you removed.

Fasten screw "B" (on the left side) next. That's one of the two middle-size screws.

For screw "A" (at the top), remember that this also attaches the lamp bracket. Use the smallest screw in this slot. Insert the lamp back into hole in the back of the button bracket, and fasten screw "A" through the lamp bracket.

Similarly, screw "D" (on the right side) attaches a wire clip, so put the wire clip back in place (hopefully with all of the wiring still threaded through) and fasten the screw through the clip. Screw "D" is one of the middle-size screws.

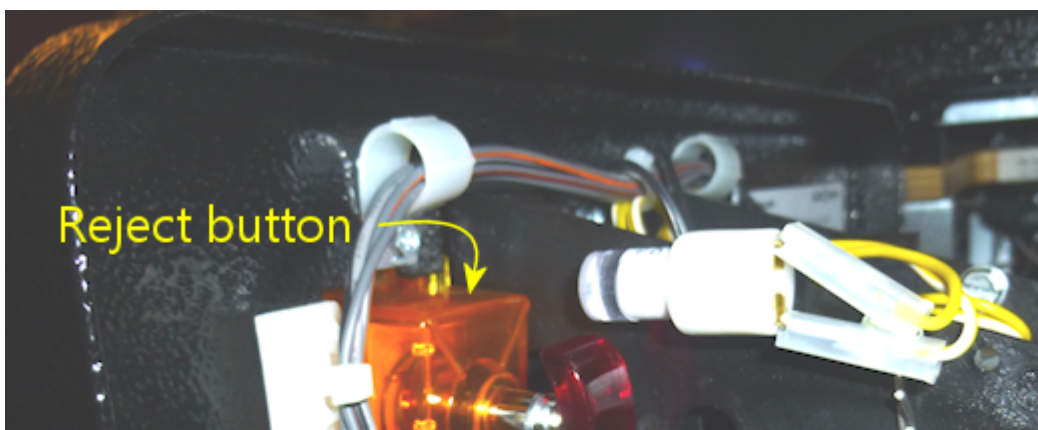
Now you just have to repeat the whole thing for the second coin chute. If it's any consolation, it should be easier the second time through, now that you've had some practice.

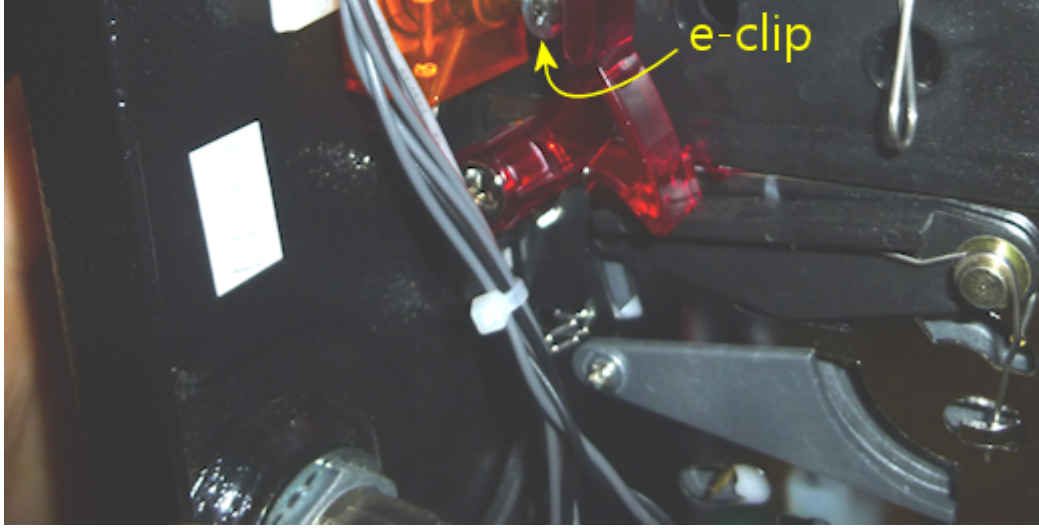
Customizing the coin button colors: The WPC coin doors come with orange plastic buttons, so you're kind of stuck with a Halloween color scheme for your inserts (orange and black). But there's a way to fix this: you can buy clear plastic replacements for the orange buttons. With clear buttons, you can use whatever color scheme you want for the printed inserts, and/or you can change the back-light color, by replacing the white #555 lamps with color LED bulbs. For the clear buttons, search for part number PBL-100-0072-00 at Pinball Life, or 27-1081-CC at Marco Specialties. Both companies also sell #555 LED bulbs in various colors.

Replacing inserts in Stern/SuzoHapp coin doors

If you opted for a SuzoHapp door instead of a WPC door, good news: they made the process of replacing the inserts quite simple in these doors - much less of a chore than for the WPC-style doors.

In the SuzoHapp doors, the orange button is held in place by an e-clip on the button stem inside the coin door. To remove the button, just pry off the clip. You might be able to get it off with fingers alone, or you can use needle-nose pliers to pull it loose.





Removing the clip frees the button and lets you pull it out of the door on the front side.



Removing the old insert is just a matter of sliding it out through the paper-thin slot in the top of the button.





There's nothing holding the insert in place except friction, but there's enough friction that it won't fall out on its own. There's a narrower slot in the bottom of the button that you can feed something super-thin into (try a folded piece of paper) to dislodge the old insert, or you might try just blowing into the bottom slot.

Once the old insert is removed, simply slip the new insert in through the top slot.

To put the button back into the door, just reverse the removal process. Put the spring back on the stem, push the button into the socket, and slip the e-clip back onto the stem on the inside. You'll probably need to use needle-nose pliers to snap the e-clip into place.

The coin door wiring harness

If you buy one of the pre-built and pre-wired coin doors, it will come with a connector plug, known in the jargon as a "wiring harness", with pins for all of the electrical connections inside the door. If you want to take full advantage of the functional components in the door, you'll have to connect the pins in the wiring harness to other points in your cab:

- Connect the switches and buttons to your key encoder. This includes the operator buttons, the coin chute switches, and the slam tilt switch, if present. All of these connect to the key encoder following the same pattern as other cabinet buttons, as explained in Chapter 35, Button Wiring.
- Connect the lamps at the top of the coin chutes to power, to make them light up when the machine is on. You can optionally connect them to your output controller to make them light up under software control; if you want to go that route, connect them just like any other feedback device, as explained in Chapter 47, Feedback Device Wiring. In my opinion, there's no good reason to go to that kind of trouble; I'd just hard-wire them directly to power. That's exactly what the real machines do, so there's no "realism" gained from controlling these through software.

The wiring harness connectors on the pre-built doors are standard parts, which means that you can buy mating connectors from an electronics vendor like Mouser. I'd recommend doing exactly that: I like using modular plug-in connectors wherever possible, because it makes the wiring neater, and it makes maintenance easier if you ever have to remove anything.

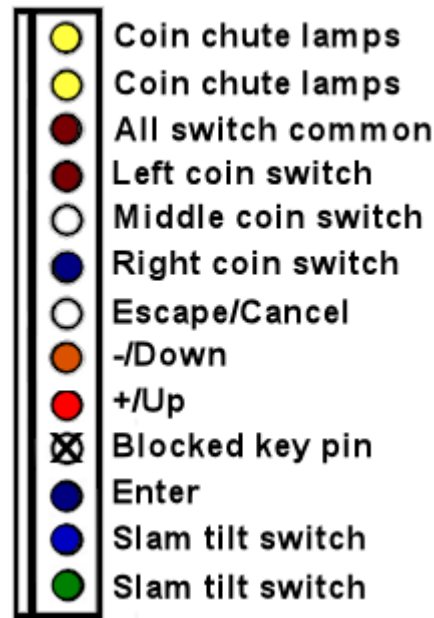
Some cab builders don't want to deal with the hassle of finding matching connectors, so they use wire cutter to snip off the wiring harness connectors and then make their connections either by soldering the wires, or using a screw-down terminal block. I don't recommend cutting off the connector, since you can't easily undo that, but some people go this route for the sake of expediency.

Whether you use the pre-attached connector or cut it off and connect the bare wires, you'll need to know how the wiring is all connected. See the sections below for wiring diagrams for the various door types.

Williams 13-pin connector

The Williams coin door comes with a pre-wired 13-pin connector of this type: Molex 09-50-3131. This connector is designed to plug into a circuit board that has the mating header: Molex 26-60-4130.

Here's how the pins in the connector are wired:



The 13-pin Williams connector. The pin marked "Blocked key pin" is physically blocked in the connector, so that the mating pin won't fit into it. The matching pin on the PCB connector must be clipped off. This ensures that you can't insert the plug backwards, or plug it into the wrong circuit board. The "Middle coin" pin isn't wired at all in coin doors for the US market, which only have two chutes (left and right). The pin colors in the diagram approximate the wire colors leading to the harness, but note that some of the wires are striped in two colors; the diagram only shows the base color.

The "All switch common" pin is daisy-chained to one terminal of each switch and button, so it corresponds exactly to the "Common" terminal on your key encoder and should be connected there. Each of the other switch pins should be wired to the corresponding button input on your key encoder.

The odd man out among the switches is the "slam tilt" wiring, which has its own separate pair of pins rather than connecting through the "common" pin. You can simply connect one of these pins (it doesn't matter which) to the "common", and connect the other to the appropriate button input for the slam tilt switch.

The coin chute lamp pins provide the 6.3V power connections for the lamps that illuminate the coin chutes. These are normally hard-wired to power on a real pinball machine, so I suggest doing the same in a virtual cab. The lamps normally installed are incandescent #555 lamps, which are designed to run on 6.3V but will run (noticeably dimmed) on 5V. Incandescent bulbs aren't polarized, so it doesn't matter which order you connect (+) and (-) on these. See Chapter 45, Power Supplies for Feedback for advice on adding 6.3V power to your cab. You can connect the coin lamps to your output controller to put them under software control if you wish, but I personally wouldn't bother: it won't add much of a special effect, since the software will just leave them on all the time anyway.

How to connect to the plug

Now that you know how the connector is wired, there's still the matter of physically connecting wires to it. There are three main options:

- Cut off the connector and use your own wiring
- Use a "Z" adaptor and wire your own matching wire harness

- Use a custom circuit board

The first option is the most straightforward, especially if you're not much into electronics as a hobby, but I don't like it much because it's not easily reversible should you want to re-purpose the coin door in the future. The Z adapter option is easy to understand, but it's really about as much work as setting up a custom circuit board, since you have to build the mating plug out of crimp pins. That's why my favorite option is the custom circuit board. That takes a little more up-front planning work, but the assembly process is easy and it's very convenient to install and use once it's assembled.

Read on for the details on how to implement each option.

Custom circuit board for Williams 13-pin plug

On the original Williams machines, the 13-pin connector plugged into the mating header on a small circuit board mounted on the left inside wall of the cabinet. This circuit board in turn had other connectors that led back to the CPU board (for the switch terminals) and the power connections (for the coin chute lamps).

You can create a simple circuit board for this, using the 13-pin header linked above. For your convenience, I've drawn up EAGLE plans for a board that includes the pin header and breaks out the connections for easy wiring to other points in your cab. You can have this board made by OSH Park for under \$10. Simply upload the EAGLE plans to their site and they'll make 3 copies of it for you.

The board design includes a 6.3V regulator to provide the correct voltage for the coin chute lamps. Just plug in a 12V supply (you can use an ATX power supply for this). The on-board 6.3V regulator can also supply power to up to four additional front panel button lamps, which usually use the same 6.3V bulbs.

Plans for the board and assembly instructions are in the appendix section Appendix 8, Coin Door Interface Board.

"Z" adapter for Williams plug

It might seem like too much work to build a whole separate circuit board just to accommodate that special coin door plug. The thing that would be a lot simpler would be to find a mating wire-harness connector that you could wire to your key encoder and power supply, and then just plug the two connectors together directly, without any extra circuit boards in between.



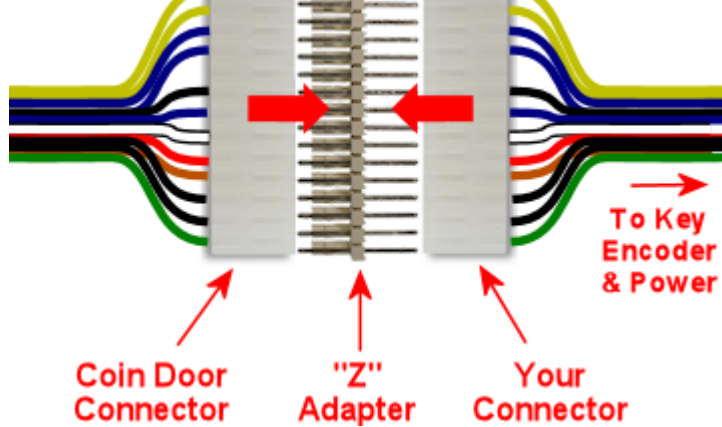
There is a way to do this, but it requires a special adapter plug that serves as a gender changer for the Molex plug, so that you can plug two of the same Molex plugs together. The adapter is often called a "Z Connector" because of its shape.

You can find these at pinball suppliers and some electronics vendors. Search for part number MWWS156-1624. Get the version with at least 13 pins (MWWS156-1624-13). A version with more pins will work if you can't find the 13-pin version.

Once you have that part in hand, you can build your own cable connector using the same connector that's on the coin door harness (Molex 09-50-3131). Then just plug the two cables together using the "Z" adapter.

This option is marginally less work than the circuit board approach above, and slightly cheaper. I don't think it's quite as tidy, since it leaves these two big cable plugs loose in the cab, but you can fix that easily enough with some cable ties.





This connector is a "crimp pin housing" type. See Chapter 82, Crimp Pins for help with ordering the necessary parts assembling the connector.

Connecting to the bare wires

The most straightforward option is to cut off the 13-pin plug and connect to the bare wires. If you're using a circuit board or a Z-adapter as described above, you can obviously skip this section. And I generally recommend you do use one of those other options, since cutting off the plug is destructive and not easily reversible, and because pluggable connectors are easier to work with later when doing maintenance. But if you don't have the patience for one of the more structured approaches above, this is a workable last resort.

One fairly easy way to deal with the bare wires after cutting off the connector is to wire them to a screw-terminal block like the one pictured at right. Look for a terminal block with 12 or more terminal positions. You can mount the block on the cabinet wall adjacent to the coin door. Install the coin door first, then connect each wire from the coin door to one of the screw terminals. Then you can run a wire from each screw terminals to the corresponding connector on your key encoder, in the case of the switch wires, or to the appropriate power connections in the case of the coin chute lamps.



Alternatively, you can create your own wiring harness using a different connector of your choice. I personally don't see a lot of benefit in this approach, since it's about the same amount of work as creating a connector that mates with the existing 13-pin plug.

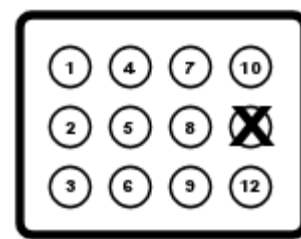
Stern SAM connector

The SAM-era coin doors use a 12-pin Molex connector for their switch wiring. Unfortunately, I haven't yet been able to confirm the gender or pin pitch - it looks like a Molex .063" plug (Molex 03-06-2122), which makes the matching connector the .063" 12-position receptacle (Molex 03-06-1122). But it could be the other way around. If you're buying one of these doors, you'll have to wait until you have it in hand to determine the exact plug type - and when you do, please let me know which it is so that I can improve the advice here!

These are connected as follows (refer to the diagram at right for pin numbering):

1. Left coin
2. Not used
3. Right coin

4. Not used
5. Not used
6. Slam tilt switch
7. Back/Cancel service button
8. Down/- service button
9. Up/+ service button
10. Enter/Select service button
11. Key (blocked pin for orientation)
12. Switch common



These doors appear to have an additional four-pin connector for power to the coin chute lamps. I don't know the type of connector used here, so again, please let me know if you have reliable information so I can include it here. At any rate, the wires you want to connect in that plug are the yellow and yellow/white wires: those connect to 6.3V power. The bulbs are incandescent, so the order of the (+) and (-) connections doesn't matter for these. You can leave the other two pins unconnected.

To integrate this connector into your own wiring, you have basically the same options as for the Williams 13-pin connector: you can either build a matching plug, or you can cut off the plug and connect the bare wires. I think it's a better idea to keep the plug for the sake of ease of maintenance down the road, but that requires a little more up-front work. Review the section on the Williams 13-pin plug above for thoughts on these options.

Stern SPIKE connector

The SPIKE-era coin door wiring harness is terminated in two connectors, each female 0.1" pin header connectors that connect to 0.1" single-row pin headers. These are common connectors that you can find easily from electronics vendors; see Chapter 80, Connectors for more on these. One of the plugs has the wiring for the service buttons, and the other has the wiring for the coin chutes and slam tilt switch.

The first connector, which has the service button wiring, fits a 6-pin, single-row 0.1" header. The pins are arranged as follows. The "Key" is a blocked pin to ensure that you insert the plug in the right direction; this pin must be snipped off on the header to allow the plug to fit.

- Escape/Cancel
- -/Down
- +/Up
- Enter
- (Key - blocked pin)
- Switch common

The second connector has the wiring for the coin chutes. This one fits a 9-pin, single-row 0.1" header. The pins on this connector are arranged as follows. There's no key pin on this connector, so you have to be careful to plug it in the right way. You can identify the end with the LED power pins by the yellow wires going to the plug.

- +5V (power to coin chute LEDs)
- 0V (power to coin chute LEDs)
- Switch Common

- Slam tilt switch
- Coin 5
- Coin 4
- Coin 3
- Coin 2
- Coin 1

Connect the coin chute LED pins to your 5V power supply. I recommend connecting these directly to the power supply, since they're similarly hard-wired to power on the real machines. But if you prefer, you can connect the 0V side to a port on your output controller to place the coin chute lights under software control. I don't feel that's worth the trouble, since the software will just leave the lights on all the time. See Chapter 47, Feedback Device Wiring for more details.

The Switch Common pins on both connectors correspond to the Common port on your key encoder and should be connected there. Connect the other button and switch pins to corresponding input ports on your key encoder.

To integrate these connectors into your own wiring, you have basically the same options as for the Williams 13-pin connector: you can either build the matching connectors, or you can cut off the plugs and use the bare wires. I think it's a better idea to keep the existing plugs for the sake of ease of maintenance down the road, but that requires a little more up-front work. Review the section on the Williams 13-pin plug above for thoughts on these options.

Creating your own wiring harness

If you buy one of the basic coin door options that requires you to install the wiring yourself, I'd recommend creating your own plug-and-socket connector for it, rather than wiring everything directly to other points in your cab. This will make future maintenance easier. The Molex .063" wire-to-wire connectors are good for this, as they come in a wide variety of pin counts. See Chapter 80, Connectors for more on building custom connectors.

Wiring to the key encoder

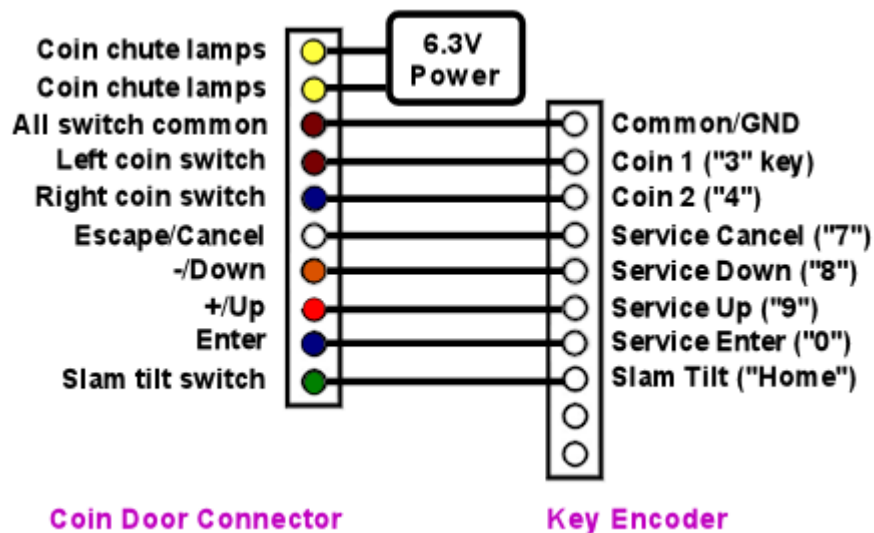
The service buttons are just like any other cabinet buttons, so wire one terminal of each button in a daisy chain to your key encoder's Common terminal, and wire the other terminal of each button to an individual button port on the encoder. See Chapter 35, Button Wiring for more on the general wiring plan for buttons.

The coin chute switches and slam tilt switch all act just like buttons, so wire them the same way.

All of the pre-wired door types provide wiring that's compatible with the general key encoder wiring plan. They all use a Common wire that's daisy chained to the various buttons; this corresponds exactly to the Common terminal on your key encoder, so simply connect the two.

What about multiple Common wires on the coin doors? Some of the pre-wired doors provide multiple Common wires for different groups of buttons. You can just connect all of the Common wires together if there are more than one. The doors that use multiple Common wires do so for compatibility with the more complex "matrix" wiring used in real machines from the 1980s and 1990s, where switches were grouped into blocks that had separate commons. The key encoders we use in virtual cabs have more modern electronics that allow a simpler scheme, with a single

common across all buttons.



Key assignments

If your key encoder is programmable, assign the coin door button ports to keyboard keys as follows. (The first four items represent the labels on the "service" buttons.)

- Cancel = **7**
- -/Down = **8**
- +/Up = **9**
- Enter = **0**
- Coins = **3** (see "How to set up multiple coin chutes" below)
- Slam tilt = **Home**

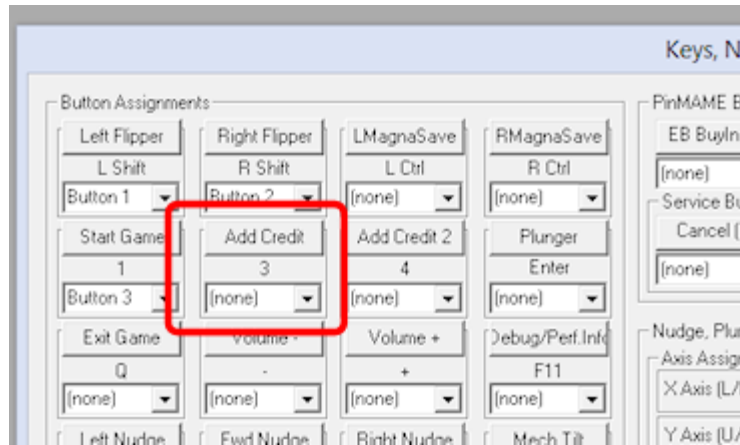
If your key encoder isn't programmable, and it uses a fixed set of joystick buttons as inputs, you'll have to do the programming in Visual Pinball instead of in the key encoder. Open the Visual Pinball editor, with no game running, and select Preferences > Keys from the menu. For each of the buttons you connected, find the entry in the dialog for that button. (The Coin buttons are labeled "Add Credit", "Add Credit 2", "Coin 3", and "Coin 4". The service buttons are grouped together in a section labeled "Service Buttons".) Under each button label, you'll find a drop list where you can select the joystick button number assigned to each input. You'll have to repeat this process with each version of VP that you intend to use on your cab (VP 9, VP X, PhysMod5, etc), since each version keeps its own separate settings.

If your key encoder isn't programmable, and it uses a fixed set of keyboard key assignments, you'll have to pick the ports on your key encoder that are already assigned to the keys listed above when wiring these buttons. If your key encoder doesn't provide some of these, you won't be able to use these buttons with that encoder. In that case, you might consider adding a Pinscape unit purely for the additional programmable key inputs. It only costs about \$15, and the procedure for wiring it for button inputs is essentially identical to the procedure used for the commercial key encoders.

Check your VP "Coin 1" setting

Open the Visual Pinball editor and bring up the keyboard preferences dialog (select Preferences > Keys from the menu). Check that the entry for **Add Credit** is the **3**. If it's not, click the **Add Credit** button, then press the **3** key on the keyboard.

While you're at it, also check that **Add Credit 2** is set to **4**, and change it as well if necessary.



*Make sure **Add Credit** is assigned to **3** and **Add Credit 2** is assigned to **4***

Note that each version of VP (9, 10, PhysMod5) stores its settings independently of the other versions, so you'll have to repeat this process for each version of VP that you intend to use on your cab.

This is required **even if you're using joystick input**, and in fact it's especially important if you're using joystick input due to a bug in the VP core scripts. Current versions of the VP scripts ignore these settings and always use **3** and **4** respectively, no matter what you set in the dialog. That's confusing at best, but it's even worse if you're using a joystick, because VP will internally map the joystick input to the key assigned in the dialog. If that doesn't match the hard-coded **3** and **4** keys used by the scripts, your coin inputs will be ignored by most tables.

How to set up multiple coin chutes

In the US, the most common coin door configuration is two slots that both accept quarters. Each slot has its own separate switch, so you'll have two inputs to connect to your key encoder. You have two options for this:

- Wire the two switches together to the same key encoder input
- Wire them to separate inputs

If both slots accept quarters, I recommend the first option: wire both switches to the same key encoder input, and assign this input to the **3** key on the keyboard. In the PC pinball simulators, this represents the primary coin slot. You can instead wire the two slots to separate inputs; if you do, use the **4** key for the second slot's key assignment, as this represents the secondary coin slot in the PC pinball software's mappings.

The reason I recommend using a single key encoder input for two quarter slots is that it simplifies the software setup considerably. Almost all Visual Pinball tables will handle the primary coin slot the same way, but the handling for secondary coin slots can vary from one table to the next. You'll have to use the operator menus to adjust pricing options on many tables if you want consistent handling for the secondary slot.

The advice above only applies if your two (or more) slots all accept the same coin denomination. If you're using slots that accept different coins (e.g., dimes and quarters), you should wire each slot to a separate key encoder input so that the software can distinguish the different coin types properly. Visual Pinball allows up to four distinct coin keys for this purpose. However, be warned that this is a bit of a

pain to set up in the software. You'll have to go into the operator menu for each individual table to set up pricing options. This is handled by the emulated ROM software, so there's no way to share the settings among games; Williams and Bally and the rest sadly lacked the foresight to build Internet cloud storage into their games in the 1980s and 1990s. What's more, each table has its own unique operator menu tree, so you'll have to learn how to use each one. The Williams games from the 1990s have nicely designed menus that are easy to navigate, but games from earlier generations can range from mildly confusing to utterly obtuse, roughly in proportion to age. Be prepared to hunt through the operator's manuals for older games.

To use two distinct coin denominations, assign the primary coin slot to the **3** key, and assign the secondary slot to **4**. You'll have to go through your installed games and set up the suitable pricing options in each one via its operator service menus.

If you're using three denominations, assign the third to the **5** key. Assign a fourth denomination to the **6** key. The PC pinball software stops at four slots, so there's not much you can do on a virtual cab with slots beyond the fourth.

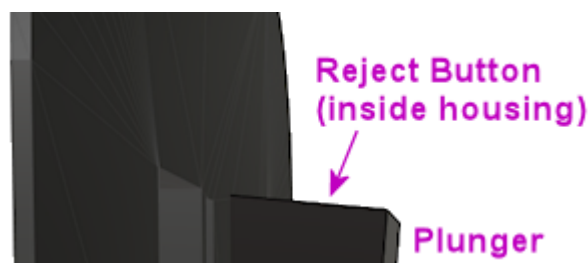
Using the Reject buttons as virtual coin buttons

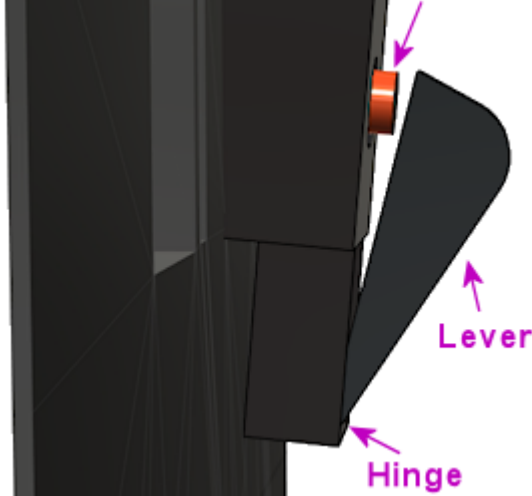
We've already mentioned that you should include some way to insert coins, either real or virtual, because of the difficulty of converting some older games to Free Play mode. If you're installing coin mechanisms in the chutes and wiring the coin switches, you'll be able to accomplish this using real coins. Alternatively, you can skip the coin mechs and just install a "Coin" button that simulates coin insertion. And in my opinion, you really should include a "Coin" button even if you install real coin mechs, since it's fun to use real coins once in a while, but only once in a while. You'll want the convenience of a front-panel button the rest of the time.

The perfect place for the "Coin" button, in my opinion, is the Coin Reject buttons at the top of the coin chutes. You could just add another pushbutton to your front panel of the same sort as the Start button, but that adds clutter and makes the machine look less "real", since real pinballs are almost all very minimalistic about buttons. The Reject buttons let you hide the Coin button in plain sight, using a button that's going to be there anyway. It's also an intuitive place for the virtual Coin button since it serves a coin-related function already. It's not realistic, of course; an arcade machine obviously isn't going to provide a button that lets anyone add free credits. But that's just all the more reason to "hide" the button like this.



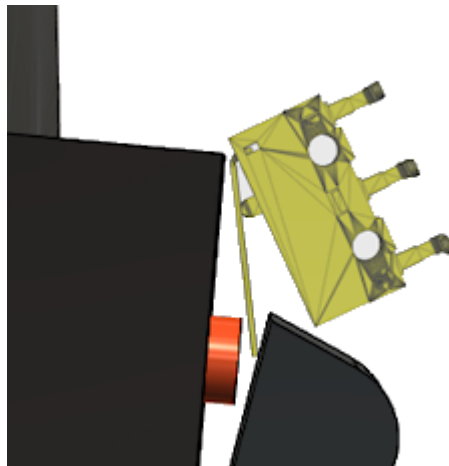
Using the Reject buttons as coin buttons isn't just a matter of wiring them to the key encoder, because there's no wiring to these buttons to begin with. They're not the electrical kind of button that operates a switch; they instead operate mechanical levers in the coin acceptor mechanisms that widen the coin passage to help free jammed coins. To make these into buttons we can wire to our key encoder, we have to add microswitches in such a way that pushing the button operates the microswitch.





Basic geometry of the coin reject button for the Williams-style coin doors (other types vary slightly in the details but have roughly the same layout). A plunger sticking out from the back of the button engages the lever when pressed. The button's total travel range is about $\frac{3}{4}$ ".

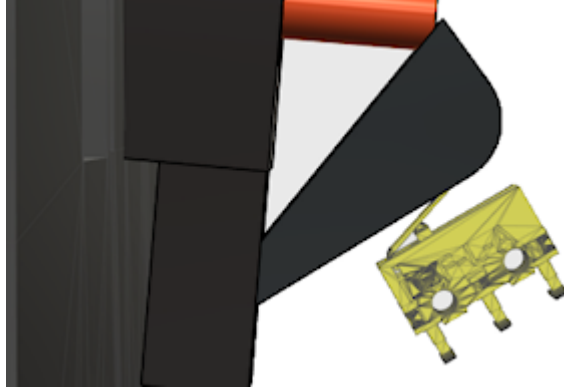
The obvious place to add a switch is just behind the plunger:



That's workable but not ideal. The problem is that the plunger has to travel about $\frac{3}{4}$ " in order to fully engage the lever that clears jams in the coin mechanism. If you put the microswitch directly behind the plunger as shown above, it'll block that full range of motion. An over-zealous player could also push the button hard enough to bend the microswitch or break it off its mounting. If you're not using real coin mechs, though, this simple arrangement might work for you, since you probably don't care about giving the button the full range needed to clear jams.

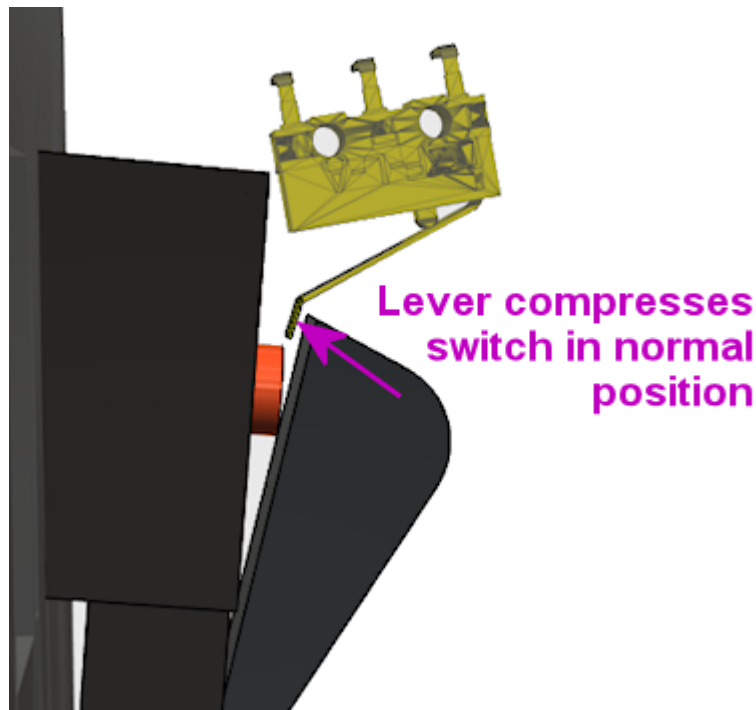
The next most obvious arrangement, if you want to avoid blocking the full travel range, is to position the switch at the *end* of the plunger's travel. There's no good way to do that in such a way that the plunger engages it directly, since as you can see, that large lever is in the way. What you can do, though, is let the lever engage the switch.





This arrangement, like the first, works but isn't great. This plan has the benefit that you *can* press the button all the way in, but it has the drawback that you *must* press it all the way in. That's unpleasant for players, because it takes a bit of effort to push the button all the way in; there's a lot of friction and spring force in the coin mech levers. It also makes the button feel very different from the other front-panel buttons, which all take a light touch.

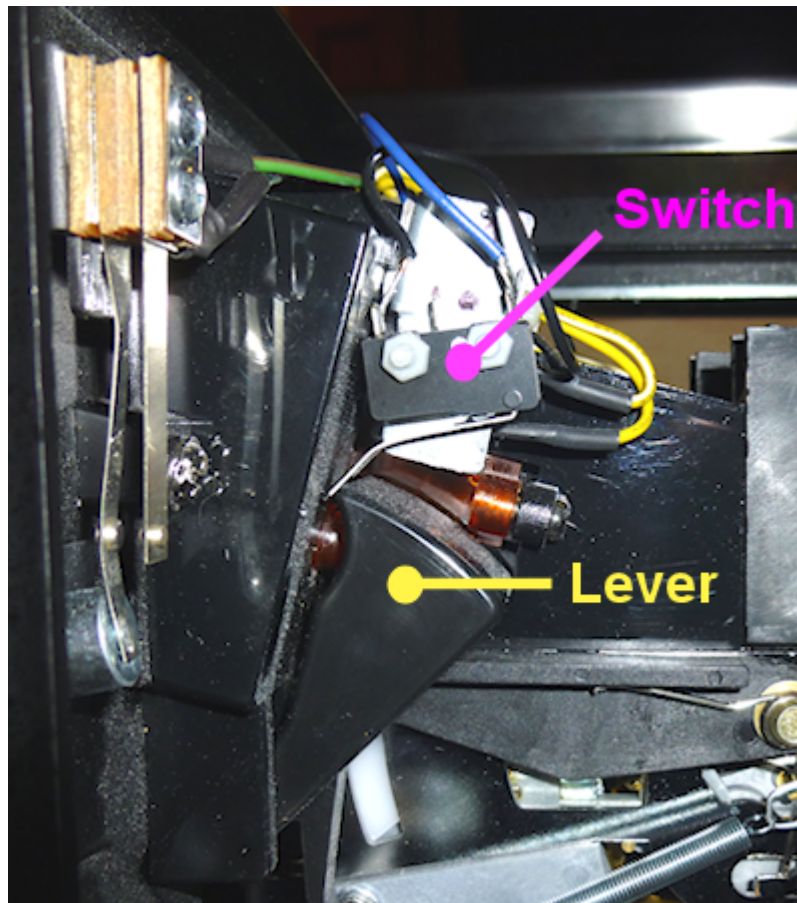
Fortunately, there's a better arrangement that overcomes both of these problems: it doesn't block the full travel range of the button, and it lets you engage the coin switch with only a slight push. The trick is to install the coin switch "backwards", so that the microswitch is compressed by the coin mech lever in the normal position, and released when the lever moves forward. It's backwards in the sense that pressing the button releases the switch, and releasing the button engages the switch.



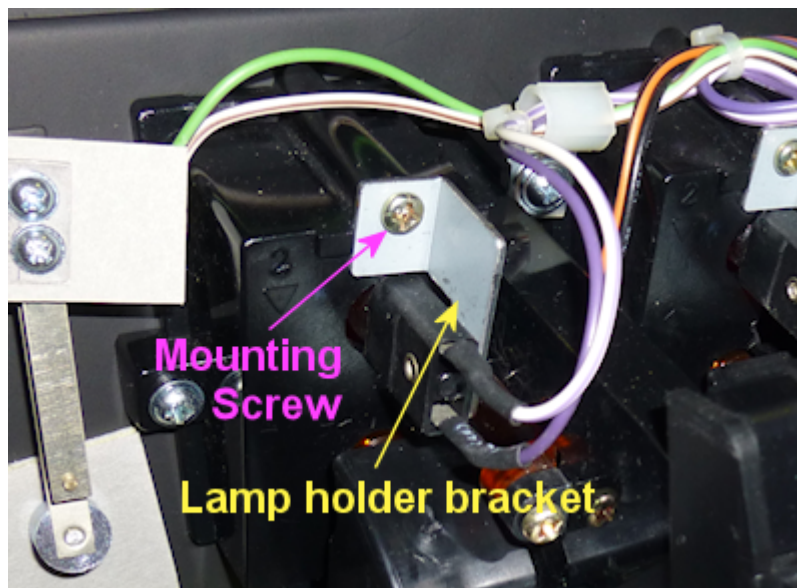
This arrangement trips the switch after pushing the button only a short distance, so it gives the button the same light touch as the other front-panel buttons. And it doesn't get in the way of the normal jam-clearing function of the button.

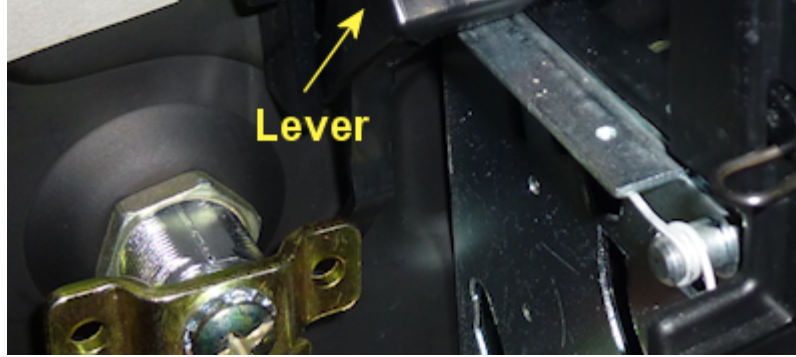
It takes a little finesse to get the positioning right. You have to position the switch so that its metal arm is held down by the big coin door lever when the button isn't being pressed. You also have to arrange it so that the switch arm stays out of the way of the plunger. This doesn't leave a lot of room to maneuver, but it can be done.

You'll have to improvise something for a mounting bracket for the switch. My recommendation is to fashion a bracket from sheet metal, or perhaps use a 3D printer to create a custom bracket with the right geometry for your switch. There's a good attachment point for whatever bracket you come up with, just above the coin chute where the lamp holder is attached. You can unscrew the lamp bracket and add your own bracket on top of it, fastening it with the same screw.



Microswitch added to Coin Reject button in "backwards" arrangement, so that the spring-loaded coin lever presses the switch when the Reject button is in its rest position. Pressing the Reject button pushes the lever forward, which releases the switch. The microswitch is mounted on an improvised sheet-metal bracket. The bracket is fastened to the door with the same mounting screw that fastens the lamp holder for the chute.





Recommended mounting point for the microswitch bracket. The coin door should have a lamp holder installed above the chute, on a bracket fastened with a machine screw at the top of the chute. If you use a similar thin metal bracket for the microswitch, you can share the mounting point. Remove the mounting screw, layer your switch bracket behind the lamp bracket, and put the screw back in through both brackets. The microswitch isn't installed in this view; this is just the "before" view to show where it goes. The purple and white wires visible here are the power wires for the lamp.

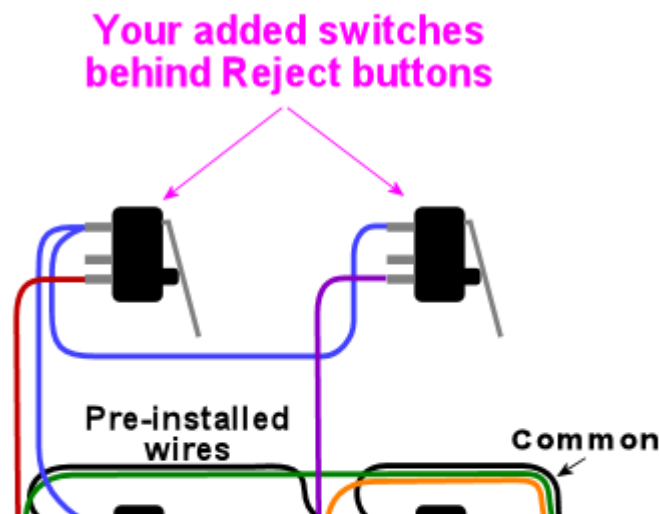
Wiring the coin switch

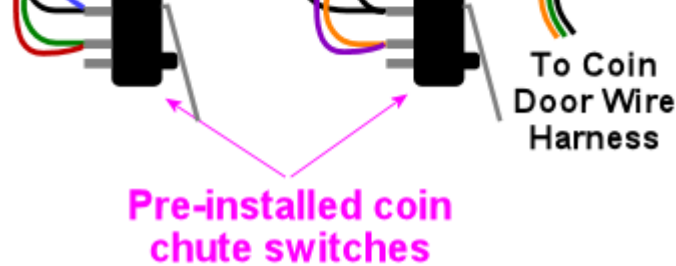
If you use the "better" plan above, where the switch is installed so that pressing the button releases the switch, the trick with wiring is to use the switch's Normally Closed terminal. This is the terminal that connects to the switch's Common terminal when the switch isn't engaged (its "normal" position). That lines up with our "backwards" switch arrangement. So with this mounting plan, simply connect the wiring to the switch's NC and COM terminals.

If you're using one of the more straightforward plans where pressing the button engages the switch, wire to the switch's NO (normally open) and COM (common) terminals instead.

My advice for the next step is to solder the two wires from the microswitch directly to the two terminals wired to the coin chute switch directly below it. That will make button switch and the coin chute switch interchangeable. They'll both end up wired to the same key encoder input, so pressing the Reject button for one of the chutes will have exactly the same effect in the software as inserting a coin into that chute.

The diagram below shows this wiring plan, with one small adjustment: we identify the existing "all switch common" wire, which is daisy-chained in the pre-installed wiring from switch to switch, and we extend that daisy chain to our new switches as well.

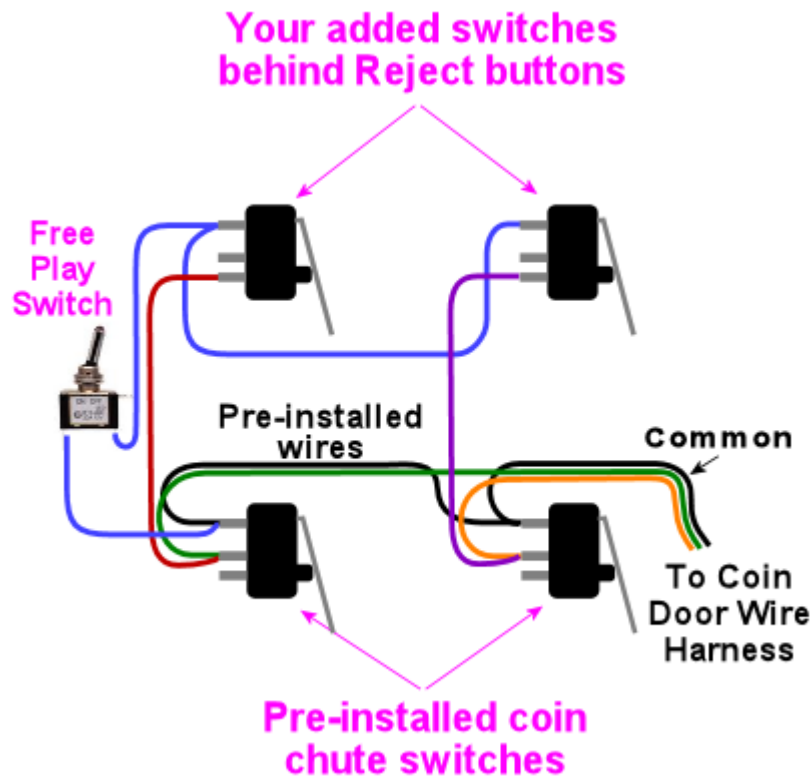




Wiring the Reject button switches. Identify the "common" wire in the existing wiring first: this is the wire that's daisy-chained between the existing coin chute switches. Extend that daisy chain to one terminal each on your new switches. Then connect the other terminal of your new switch to the corresponding other terminal of the existing coin chute switch.

Wiring a Free Play mode switch

If you want to get really fancy, you can add a switch that enables or disables the Reject button switches. That will let you turn Free Play mode on and off as desired. This requires only a small change the basic wiring plan above: splice a switch into the "common" daisy chain wire before your first added switch. When the switch is off, this will disconnect both of your added switches from the common wire, so they'll do nothing when pressed. When the switch is on, the common connection is restored, so they'll work as normal.



Other places to hide buttons in the coin door

Many cab builders these days prefer to minimize the number of visible buttons on the front face, to better replicate the uncluttered look of a real pinball machine. You might like the aesthetics of fewer buttons, but sometimes you do want a few extra buttons for the sake of functionality and ease of use. One way to resolve these conflicting goals is to create "secret" buttons that function like regular pushbuttons but don't look like buttons. We saw above how you can do this with the Coin Reject buttons, taking a standard feature of the coin door that's there anyway, and turning

it into a pin cab pushbutton - you get a functioning extra button without any added visual clutter. The coin door offers a few other opportunities for secret buttons:

Coin return slot flaps: If you don't plan to implement working coin chutes, you can use the little flaps in front of the coin return chutes as secret pushbuttons. Just install a microswitch behind each flap, so that pushing the flap operates the switch. (This won't work if you do want working coin chutes, since you'll want to leave the return chutes clear, so that you can remove rejected coins as usual.)

Coin door lock: If you're using an entirely fake coin door, it'll probably have a fake lock tumbler, to simulate the appearance of the real doors. If that's removable or made of flexible plastic, you could position a microswitch behind it and use it as a button.

Dollar bill acceptor plate: All of the modern coin doors for the US market (including the WPC, Stern, and SuzoHapp models) include removable rectangular plates where you can install a dollar bill acceptor mechanism. The space behind these plates on the inside of the door is always left open, to leave room for the bill reader, for operators who install them. I've never heard of anyone installing a dollar bill reader in a home-use pin cab, so this space usually goes unused. You could either drill one or more holes in the plate and install pushbuttons, or replace the plate with your own button panel. This isn't really hidden the way the other ideas above are, but you can at least make it inconspicuous by using low-profile black pushbuttons. Alternatively, with a little improvisation, you could make the cover plate itself into a button by attaching a hinge on one side and placing a switch behind the plate.

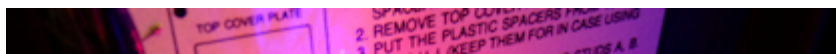
Adding an extra service panel

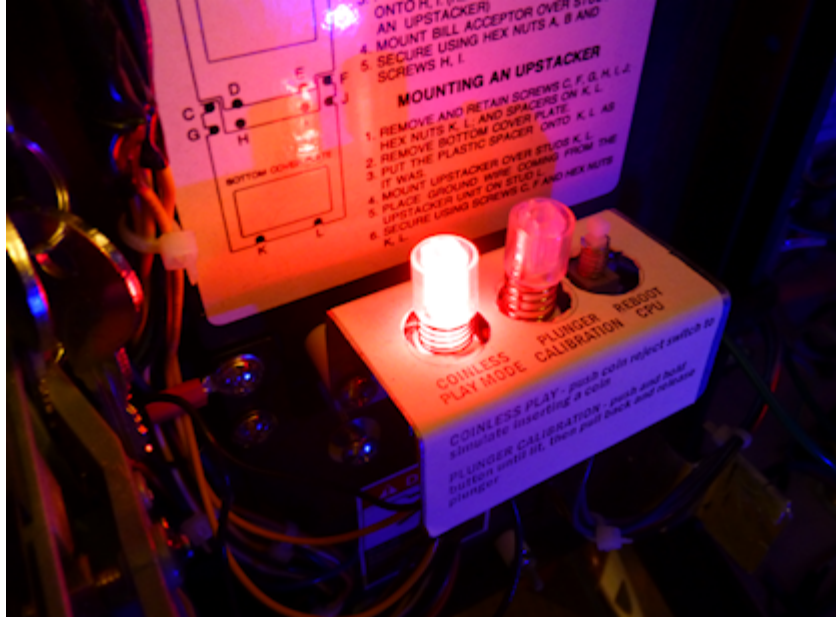
The inside of the coin door is a good place for any extra "operator" controls you need in your cab. I wouldn't put anything you might access frequently here, but it's great for controls you use occasionally, since it avoids adding any visible clutter to the exterior of the cab, but is still in easy reach when you need them. Some examples of suitable controls:

- Night Mode switch
- PC hard-reset button
- Plunger calibration button
- Master volume knob for your audio amplifier
- Audio mute
- TV power controls (in case you have a balky TV that doesn't always power on at the right times, or that needs to be power cycled occasionally because it loses the signal from the PC video card, like my DMD monitor occasionally does)

Most of the modern US coin doors have a large open space next to the coin chutes where you can install a dollar bill acceptor, and include mounting screws in that area. That makes a good place to mount an additional set of buttons.

You can buy an extra operator button panel of the same sort that's already installed in the WPC doors, and install it in the dollar bill reader area. Those control panels are just rows of ordinary buttons, so they're easy to wire for any control that uses a momentary switch. These are also fairly easy to improvise using individual pushbuttons and a sheet-metal bracket.



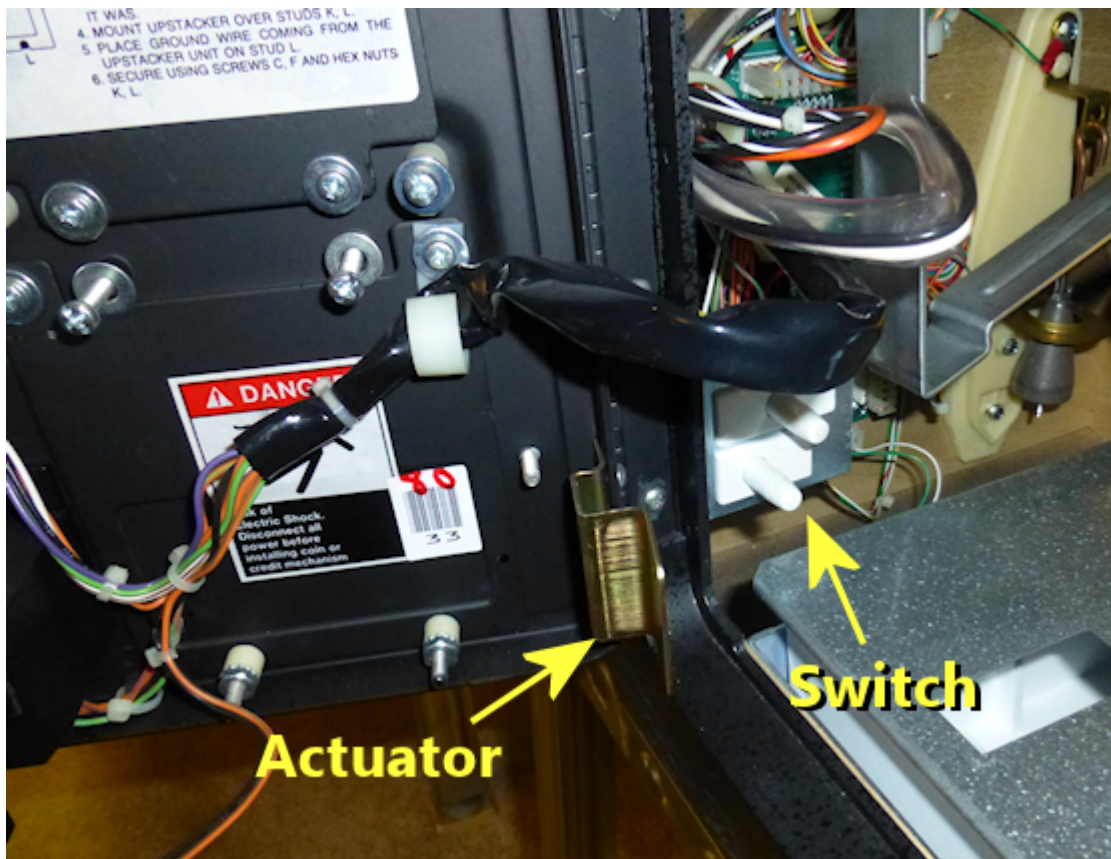


The extra service buttons in my pin cab, inside the coin door, in the space provided for a dollar bill acceptor.

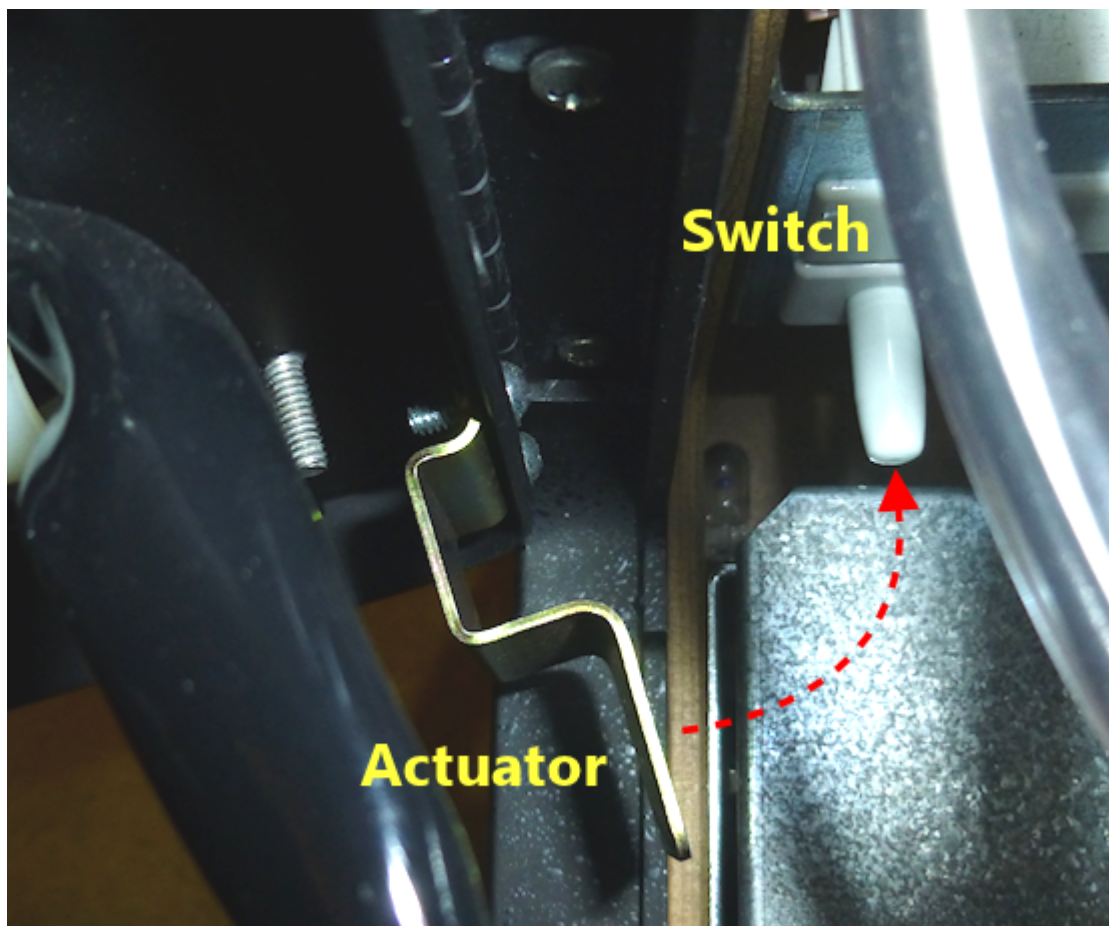
Coin door position switch

On real pinball machines, there's switch inside the coin door that detects when the door is open and closed. You'll probably want to install one of these on your virtual cab, for reasons we'll come to shortly.

The coin door switch works like the light switch in a refrigerator door. The switch is located just inside the door at the bottom of the hinge. A metal actuator plate attached to the door presses down on the switch's plunger when the door is closed.



Above: Coin door switch on a 1990s Williams machine. These machines typically use two switches stacked vertically; you can see the individual switch plungers. One switch is the high-voltage power supply safety interlock, which cuts power to the playfield solenoids when the door is open. The other switch is a logic input to the CPU that lets the ROM software detect when the door is open.



Top view of the switch and actuator, to make the geometry easier to see. The "actuator" is just a fixed metal plate attached to the door that presses down on the switch plungers when the door is closed.

The switch is a safety interlock on the original arcade machines. The 50V power line that supplies voltage to the big solenoids on the playfield is routed through this switch, which disables the 50V feed as soon as the door is opened. That helps avoid electric shock when working inside the machine.

The ROM software on machines from the 1990s and later also monitors the switch. When you open the door, the software typically does three things:

- Displays a message to let the operator know that the high-voltage power has been turned off
- Enables access to the game's setup menus through the operator buttons
- Enables write access to the game's non-volatile memory, so that settings changes made through the setup menus can be saved

Which brings us to the reason to include a coin door switch in your virtual cab. When you're running a re-creation of a modern game, you'll have to send the "coin door is open" signal to the VPinMAME emulator before the software will allow access to the menus. That might seem like a ridiculous restriction to replicate in a simulated

version, but remember, we're usually running the original ROM software, so *everything* in the software works just like on the real machines. Even odd things like this.

If you really want to avoid installing a coin door switch, there are ways to work around the ROM warnings, such as pressing keys on the keyboard to simulate the door switch. But why bother? Installing the switch is pretty easy and cheap, and it makes things work exactly like they're supposed to. I think it's easier in the long run to set it up like the software expects.



The answer to the virtual pin cab FAQ, "Why do I need a coin door switch?" With many games from the DMD era, a message like this will appear on the DMD if you try to use the operator menu buttons while the coin door is closed. This is a safety precaution on the real machines, but it carries over to virtual cabs because we're running the same ROM software.

Toggle mode or switch mode

The **old, bad** way of handling the coin door switch was as a "toggle" button. That is, each time you opened or closed the *real* coin door, you had to send a key press to Visual Pinball to toggle the open/closed status of its *simulated* coin door.

Up until recently, that was VP's only way of handling the coin door switch. This comes from VP's desktop heritage. In desktop play, you don't have an actual coin door with a switch; you do everything with a keyboard and mouse. So VP gave desktop users an easy way to simulate an imaginary coin door: press a key to open the imaginary door, and press the same key again to close it.

Needless to say, this desktop/keyboard scheme made things tricky for cabinet builders with real coin doors. The "natural" way for a door switch to operate is to be ON or OFF according to whether the door is open or closed. There's no simple way for a switch to "press a key" whenever it *changes* between ON and OFF, which is what the old VP model asked it to do.

Fortunately, VP now has a option that lets cab builders wire their coin door switches the natural way. You don't have to worry about the old "toggle" scheme any longer. The installation and wiring instructions below are all designed with the new and better system in mind. You'll have to do a little setup work in VP to enable the new "switch mode", since the old "toggle mode" is still the default, but it's fairly easy (certainly easier than setting up the physical switch as a toggle control), and we'll explain everything below.

If you read forum posts about the coin door switch, be warned that you'll see a lot of material about setting up toggle-mode switches, and it might make you worry that we're missing something important: everyone on the forums says it's so hard! My advice is to just keep reminding yourself that all of that is old and obsolete advice, from back in castle times when toggle-mode was the only option and you had to get your Internet on AOL dial-up. If you follow the instructions below, the result will not only work, but will work more reliably than the old toggle schemes did.

Recommended parts

In your virtual cab, you'll wire this switch to your key encoder, just like any other button (see Chapter 35, Button Wiring). This means that you can use almost any type of switch or button. Many people improvise with an ordinary microswitch, for example. My recommendation is to do it the easy way and use the same parts they use in the real machines:

- Switch: Williams part no. A-18249-1, A-18249-3; Stern 180-5136-00
- Mounting bracket: Williams 01-12676

You can search for those part numbers at any pinball or arcade supplier to find the correct parts. You can also find equivalent switches under these part numbers: Cherry E75-E79, Lamp PP2-1H7-2A2.

You can also use an ordinary microswitch, but I find those harder to set up for this situation, because they have very short throws that require pretty precise alignment. The plunger switches are much easier to set up for this, especially if you're using the purpose-built mounting bracket, mostly because they have nice long throws that give you lots of leeway in aligning them.

Installing the switch

You should install your coin door before installing the switch, so that you can use the coin door as the alignment guide. You'll also want to remove the playfield TV or lift it up so that you can access the inside of the coin door area from above, with the coin door closed.



Start by installing the switch in bracket, as shown at right. The rotation of the switch doesn't matter.

Close and lock the coin door. Position the switch so that the plunger is pressed against the actuator plate on the coin door, and pressed down most of the way so that the switch is in the "on" position. It doesn't have to be pressed in 100%, but it should be pressed in most of the way, to ensure that the actuator will reliably engage the switch every time you close the door.

Once you have a good position mapped out, hold the plate in place while you mark the locations of the screw holes on the front wall of the cab.

Remove the switch assembly and drill some small pilot holes at the screw locations you marked. (Be sure not to drill more than 1/2" deep so that you don't drill through the front face!) I recommend using 1/2" #6 wood screws for this (and most other things that you need to attach to the inside cabinet walls). Remove the switch from the plate, and screw the plate securely to the wall using your pilot holes as a guide.

The plate should now be positioned properly, so you just have to pop the switch back in. Before you do, though, you might want to connect the wires, since that's easier while the switch is dismantled.

Once you're ready to re-mount the switch, slip it back into the mounting plate opening, then test that the switch clicks on and off when you open and close the door.

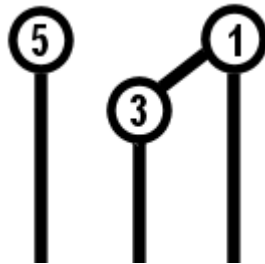
Wiring the door switch

If you're using one of the big "plunger" switches recommended above, it might look

a bit confusing, since it'll have six terminals. It's easier than it looks, though, because we only have to connect wires to two of the terminals: the Common (COM) and the Normally Closed (NC) for one of the two switches inside the device. For the Lamb and Cherry switches listed above, use these two terminals:



Check the labeling on your switch to make sure it matches this terminal layout. On the Lamb switches, they usually print the legends COM1, NC1, NO1, COM2, NC2, NO2. Other switches might use a notation like this:

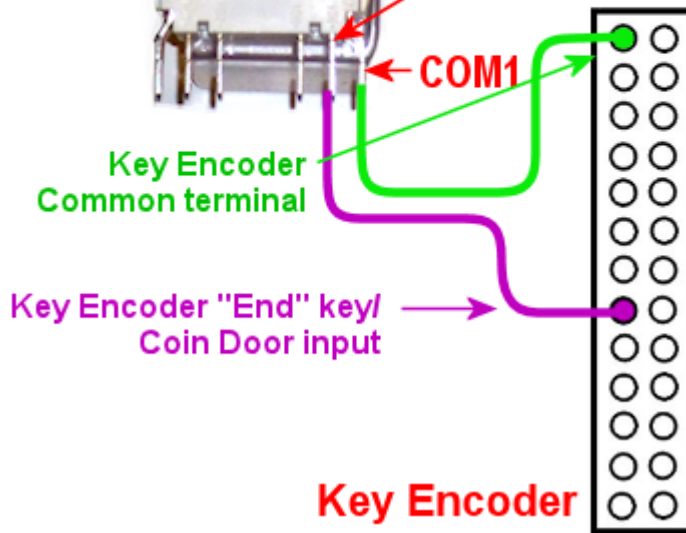


*Labeling on some switches of this type. The numbered circles represent the switch contacts; the vertical lines lead out to the corresponding external terminals. The diagonal line shows the "Normally Closed" (NC) connection, that is, the terminals that are connected when the switch is at rest, with the plunger not depressed. The other terminal is the "Normally Open" (NO) terminal. This type of diagram can be ambiguous because it doesn't show clearly which terminal is the "Common" and which is the NC. In the case of these plunger switches, the **outside** terminal is typically the Common. So the correspondence with the COM/NC/NO labeling format is 1=COM, 3=NC, 5=NO.*

The terminal layout for the Cherry and Lamb switches listed above is the same, but it's always a good idea to check the labeling to be sure your switch matches. You can double-check with a multimeter in continuity test mode if you're not sure how to read the labeling.

Once you've identified the COM and NC terminals to use, wire your switch to your button encoder like this:





Why use the Normally Closed side? Don't people usually use Normally Open when wiring a switch? Yes, that would be more typical, but in this case VP lets you wire it either way, and our VP setup instructions below assume you're wiring to NC. Given that it's all the same to VP, we suggest wiring to NC because of the effect it will have on your regular keyboard input. Wired this way, the key encoder input will be OFF whenever the coin door is closed, which it probably will be most of the time. That means the key encoder won't be sending any key presses on behalf of the coin door as long as the door stays closed. When you open the door, it will return the switch to its default position, connecting the NC side of the switch and turning the key encoder input ON. So your key encoder will tell the PC that you're holding down the End key whenever the coin door is open. When running VP, this won't make any difference. But when you're *not* running VP, wouldn't you rather *not* have the key encoder sending the extra End key press all the time, so that you can use the End key on the keyboard for text editing and other functions? That's why we suggest wiring things this way: as long as the coin door is closed, the coin door won't cause any extra key presses to register on the PC.

How to choose which key encoder input to use

- If your key encoder has a special dedicated input for "Coin Door", use that
- If your key encoder has a special dedicated input for the "End" key, use that
- If your key encoder is programmable, use any free input. Run the setup program for your key encoder (e.g., for Pinscape, run the Pinscape Config Tool; for i-Pac, use the i-Pac setup program). Find the input that you wired the switch to. Set this input to send the keyboard **End** key to the PC.
- If your key encoder isn't programmable, and uses joystick buttons as inputs, use any free input, and note the **joystick button number** assigned to that input. Run Visual Pinball - don't load or run a table; just open the blank editor. (Click Cancel if VP shows an Open File dialog.) Select Preferences > Keyboard in VP 9 or "Keys, Nudge, and DOF" in VP X. Find the section in the middle labeled "PinMAME Buttons", and find the entry for "Door (END)". Underneath that label, use the drop list to select the joystick button number assigned to the key encoder port you selected. Note that you'll have to repeat this process with each version of VP (9, 10, PhysMod5, etc) that you intend to use, since they all keep their own independent settings.

Setting up the door switch in VP

As described earlier, Visual Pinball is programmed by default for desktop users who use the keyboard to simulate an imaginary coin door, by pressing the End key on the keyboard to "toggle" the door's open/closed status. That's no good for us, since we're using a real coin door with a real position switch. Fortunately, VP can handle this situation as well, but we have to specifically tell it to override the defaults.

If you're already read through Chapter 15, Pinball Software Setup, you might recall my advice to set up a "customization log" - a text file on your cab's desktop where you record special changes you make to your VP setup, so that you can repeat them the next time you update to a new VP version. This is one of those situations! Make a note of the changes you make here.

- Make sure no VP windows are open
- In the Windows desktop, go to your Visual Pinball installation folder
- Open the Script sub-folder
- Find the file VPMKeys.vbs. Right-click it and select **Edit** from the menu. (If that doesn't work for some reason, open Notepad, then select File > Open from the menu to open this file.)
- Find the following lines:

```
toggleKeyCoinDoor = True           ' If true then...
inverseKeyCoinDoor = False         ' If false then...
```

- Change those lines to:

```
toggleKeyCoinDoor = False          ' If true then...
inverseKeyCoinDoor = True           ' If false then...
```

- Save changes (Ctrl+S)

The next time you run VP, it will read this script when you run a table, and the table should respect these keyboard settings for handling the coin door switch input. Note that these scripts are shared by all versions of VP installed in this same folder, so you don't have to do anything extra for the different versions.

Setting **toggleKeyCoinDoor = False** changes from the default "toggle mode" that desktop players use to the switch mode that cabinets use.

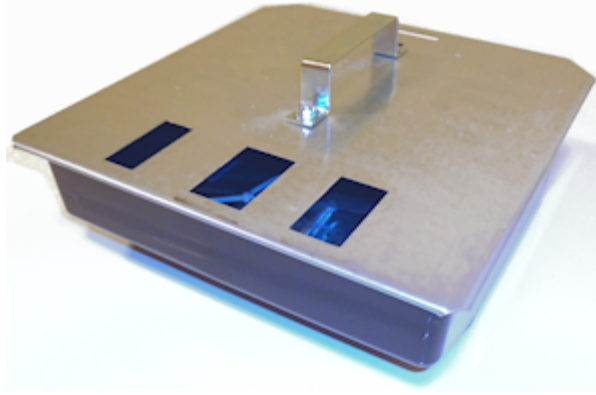
inverseKeyCoinDoor = True tells VP you're using the "Normally Closed" wiring recommended above in Wiring the coin door switch: it tells VP that the key is pressed when the door is open. (I wish they'd chosen more intuitive variable names for this!) If for some reason you prefer to use "Normally Open" wiring for your switch, leave the default (False) setting in effect for this one.

What about Future Pinball or others? As far as I know, VP is the only available pinball emulator that currently uses emulated ROMs, which makes it the only one where the coin door switch is used at all. All of this extra setup should therefore be irrelevant to other pinball programs.

Cashbox

If you install coin acceptor mechanisms in your coin chutes, you should install a container under the chutes to capture inserted coins. You don't want them bouncing around loose inside the cabinet where they could come into contact with wiring or make other mischief.

The easiest solution is to use a real pinball cashbox, like the one pictured at right. There's a standard design for these, with slots in the lid that align with the coin chutes. See the "Cashbox" sections in Chapter 21, Cabinet Body and Chapter 23, Cabinet Hardware Installation for more on how to install these.



The only downside of the standard cashbox is that it's rather large: about 10.5" wide and 11" deep. That consumes a lot of floor space at the front of the cab that you might prefer to use for PC parts, fans, or feedback devices.

If you want something more compact, you can improvise a smaller box using wood, plastic, or cardboard. For my own cab, I didn't have enough space for the regular cashbox, so I found a rectangular plastic food container of about the right height, and used an X-acto knife to cut slots in the lid that line up with the coin chutes. A bungee cord hooked to a eyelets in the cab floor holds it in place. It's not an elegant solution, but it doesn't have to look nice given that it's hidden inside the machine.

41. Audio Systems

In this chapter, we'll look at how to design and implement your virtual cab's sound system. We'll help you decide on an overall system design, as there are several ways to set this up, then we'll get into the details of what equipment you can buy and how to install and configure it.

Visual Pinball's two soundtracks

Before we start on the equipment, it's worth taking a moment to understand the way Visual Pinball generates sound effects. It has some special features for virtual cabs that are a little beyond the basic task of playing back the game's music.

Visual Pinball provides *two* soundtracks for each game:

- The "music" soundtrack. For any table that's a simulation of a real pinball machine, this is the soundtrack from the game's original ROM software. It includes all of the synthesizer music, voice cues, and other digitized effects that were played back through the speakers on the original machine.
- The "mechanical" soundtrack. This includes all of the sound effects that *aren't* from the original ROM - the ones that are added into the Visual Pinball simulation. These are called "mechanical" effects because they're almost always the sounds of things happening on the playfield: the ball rolling across the playfield and bumping into things, flippers, bumpers, and slingshots, chimes and bells in EM machines, and anything else in the physical game.

The point of separating the two types of effects into separate soundtracks is that it lets you play them back through separate speakers. In a pin cab, this means you can have VP play the mechanical sound effects through a separate set of speakers hidden inside the main cabinet, under the TV, so that the playfield effects sound like they're actually coming from the playfield. This adds to the illusion that it's a physical pinball game.

Newer versions of VP take the idea one step further with a "surround" option. This lets you use *four* speakers in the cab, so that VP can adjust the relative volume levels to make it sound like each effect is coming from a precise position in space. Mechanical effects don't just sound like they're coming from the playfield, but sound like they're coming from a precise position on the playfield.

VP doesn't require that you provide all of these speakers. If you only have two speakers, VP will mix all of the effects together and play them back on your two speakers. That's in fact what it does by default; you have to go a little out of your way to achieve the surround effects. So the multiple soundtracks are a bonus feature that you can take advantage of if you have the extra equipment, but which you can just ignore if not.

As far as I know, Visual Pinball is the only pinball simulator with this notion of separate "music" and "mechanical" effects channels. All of the other emulators will just provide a single soundtrack, with all effects playing back through your main speakers. If you don't plan to run Visual Pinball, you probably wouldn't want to bother with the extra surround speakers.

Removing unwanted VP sound effects for DOF solenoids

If you're using solenoid-based feedback devices on your cabinet to simulate flippers, kickers, bumpers, and so on, you probably **won't** want to hear VP's recorded sound effects for the same devices. The recorded sound effects tend to sound artificial

compared to the real thing, and they also sound jarringly redundant when they fire at the same time as the real devices.

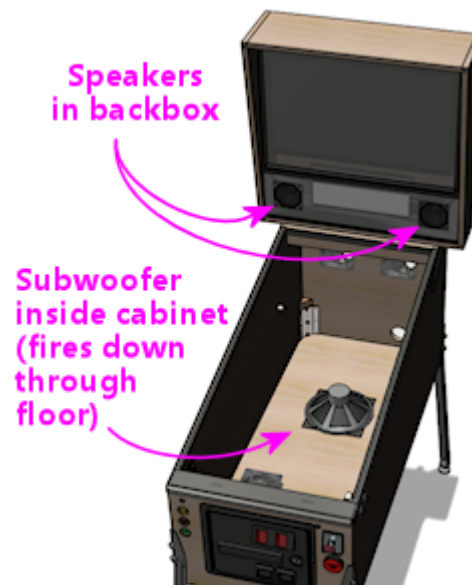
Fortunately, it's possible to disable individual mechanical sound effects in Visual Pinball. The procedure is detailed in "Disabling unwanted sound effects in a VP table" in Chapter 46, DOF Setup.

Sound system architecture

If you want to take advantage of VP's multiple soundtracks, you need to provide the extra speakers it uses. If you want to keep things simpler and/or cheaper, you can omit those. With that in mind, there are basically three ways to set up pin cab audio system:

- **Basic stereo:** The simplest setup is a stereo system, with two main speakers and an optional subwoofer. The main speakers are usually placed in the backbox, facing the player, and the subwoofer is inside the cabinet, facing down through an opening in the floor.

Most people use the same speaker placement that the real machines used in the 1980s and 90s (and that newer titles still use today, for the most part), with the main speakers mounted on a separate panel at the bottom of the backbox. With this setup, all sound effects (including the playfield mechanical effects) are played back through this single set of speakers.

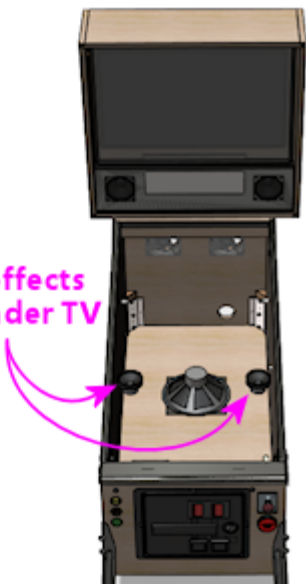


To implement this, you need:

- The two backbox speakers, typically 4" or 5.25" full-range car speakers
- The subwoofer, typically a 6" to 7" car speaker
- A 2.1-channel amplifier (two main channels plus a subwoofer channel)

A separate sound card isn't needed as long as your PC motherboard has built-in audio.

- **Music speakers plus separate monophonic playfield speakers:** This is the next step up. We keep the backbox speakers and subwoofer from the basic stereo system, and add one or two speakers inside the cabinet, under the TV. These are for the playfield effects. The added in-cab speakers should be placed somewhere around the middle of the cabinet.



Playfield effects
speakers under TV

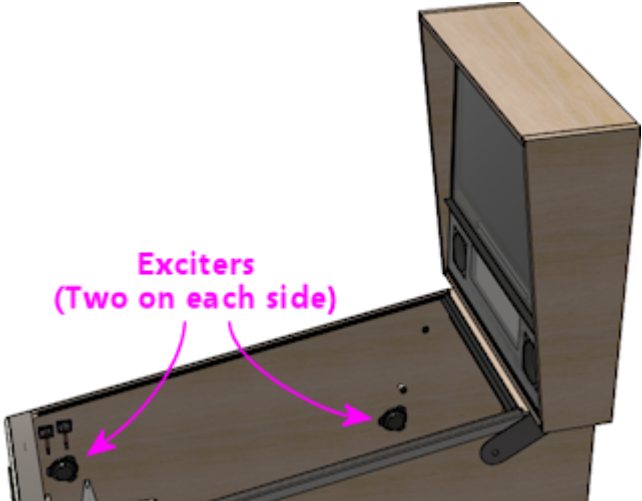
This setup makes mechanical effects sound like they're coming from the playfield, but they all sound like they're coming from the middle of the playfield, with no attempt to place them at different positions spatially. VP will only play back monophonic playfield effects with this setup, even if you have two speakers, because VP requires four speakers if you want to use the full spatial placement feature.

To implement this, you need everything from the basic system, plus:

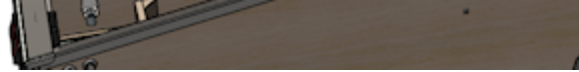
- The in-cab speaker or speakers, typically small speakers or "exciters"
- Another 2-channel or 2.1-channel amplifier
- A second sound card, only if your PC motherboard doesn't have at least 5.1-channel surround sound audio output

I wouldn't recommend this approach for a new build, as the four-speaker set up is not much more work to implement and provides much greater functionality. This one- or two-speaker setup is really only something you would have before VP had the surround sound ability.

- **Separate music speakers and playfield surround speakers:** This is the most powerful setup. We keep the backbox speakers and subwoofer from the basic stereo system, and we add *four* speakers inside the cabinet, placed near the corners. This lets VP balance the volume among the four in-cab speakers to make it sound like each mechanical effect is coming from a precise position on the playfield.



Exciters
(Two on each side)



To implement this, you need everything from the basic system, plus:

- Four in-cab speakers, typically "exciters"
- Two additional 2-channel or 2.1-channel amplifiers, or a 4-channel amplifier
- A second sound card, only if your PC motherboard doesn't have 7.1-channel surround sound audio output

This is the setup I'd recommend, if you have the budget for the extra equipment. Separating the mechanical effects and playing them back through in-cab speakers makes a subtle but noticeable difference. This isn't the most dramatic upgrade you can make, so I wouldn't prioritize it above, say, some of my favorite tactile devices (flashers, shakers, fans, replay knockers). But it's definitely a nice upgrade for a well-equipped cab.

Sound cards

In the past, PC audio usually required adding a sound card. These days, that's usually not necessary, since most PC motherboards now include built-in audio, usually with full surround-sound capabilities.

I think the only reason to add an extra sound card would be if your motherboard audio doesn't support 7.1 surround sound, and even then, you'd only need a separate sound card if you want to use Visual Pinball's full surround capabilities. In that case, you'd need an add-in card with 7.1 support. Like anything else in the PC world, sound cards come in a wide range of prices, from dirt-cheap to extravagantly expensive. The high-end cards are marketed especially to gamers, so you might be tempted to buy a deluxe gaming sound card, given that a pin cab is a sort of gaming rig, but I don't actually think it's worth the extra money. Remember that a lot of the original sound effects in pinball simulations come from games that were built with 8-bit hardware in the 80s and 90s, so most of the source material isn't all that demanding. In my opinion, any decent modern sound card will be more than up to the task. I'd just look for an inexpensive 7.1 card that gets decent user reviews on Amazon or NewEgg.

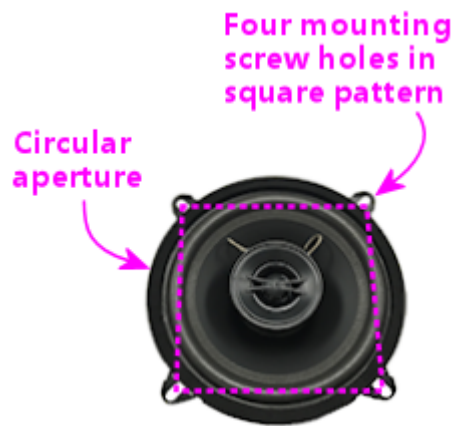
A lot of people worry that Windows will get confused if you add a sound card in addition to the built-in audio. Happily, this is one of the rare cases where Windows makes things easy. Windows is perfectly happy to have multiple sound cards installed. There should be no complications if you do decide to add one.

Main speakers

Equipment: For the two speakers in the backbox, most people use car speakers. If you're planning to use a 1990s style speaker/DMD panel, and you've already purchased or built it, it'll have cutouts for a particular size of speaker - either 4" or 5.25" - so you'll need to match that size.

The 4" and 5.25" size specs are standard car speaker sizes. These are "nominal" sizes that refer to the cutout size needed. If you shop on Amazon or elsewhere for car speakers, most products will conform to the standard sizes. There are lots of other standard sizes and shapes - what you're looking for is the circular type, with a circular aperture and a set of four mounting holes in a square pattern around the

perimeter. It should be obvious from the pictures when shopping.



The subwoofer is less constrained on size. Anything from 6" to 8" with a round aperture should work. You could even go larger, but remember that space in the cab will be somewhat limited when you have everything installed.

If you want something more targeted for pinball, Flipper Fidelity makes a number of speakers specifically designed for real pinball machines. They'll fit a virtual cab just as well if you're using one of the standard 1990s-style speaker panels. The Flipper Fidelity speakers are similar in design to car speakers, so you can find other options at lower prices with something more generic, but Flipper Fidelity's products save you some legwork in that you can be assured that they'll be the right size and that they'll sound decent in a pin cab setting. I've purchased some of their speakers myself, and I think they're well designed and sound good.

In my opinion, it's hard to go too wrong on the speakers as long as you find something of reasonable quality in the right size. There are lots of good car speakers on the market, and while some are certainly better than others, I think any speaker that gets positive user reviews from people using them in cars will also sound good in a pin cab.

PC speakers: Some people use PC speakers instead of separate components (like car speakers). PC speakers have the advantage that they come with their own built-in amplifiers, so you can just plug them straight into your PC's audio jacks. With component speakers, you have to install a separate amplifier (which we'll come to shortly).

The downside of PC speakers is that they're built into little black plastic boxes that are meant to be placed on a desktop or attached to a monitor. It can be difficult to make these look properly integrated with a pin cab, especially if you're using one of the standard 1990s speaker panel designs. The standard speaker panels are specifically designed to accommodate car-type speakers, so those integrate easily. You might be able to improvise something with a PC speaker, but it's not a natural fit. It is possible, though - some cab builders have successfully used this approach. In addition, if you don't care about integrating the speakers, you can always just pop them on top of the backbox or something like that. I wouldn't be happy with that kind of setup aesthetically, but it would be perfectly functionally.

Placement: The main speakers are usually situated in the backbox, facing the player. If you're using a 1990s-style speaker/DMD panel, you already have the natural place for them. If not, you'll have to come up with your own ideas for where to put them. For the best sound, I'd orient them so that they're facing the player, and put them somewhere in the backbox, with circular openings about the size of their apertures. They'll sound a bit muffled if you put them behind solid plywood

without the openings, and the sound might seem to come from odd directions if you don't have the speakers facing the player.

The subwoofer typically goes inside the main cabinet, on the floor, with its speaker cone pointed down. You'll need to cut a circular hole in the cabinet floor about the same size as the speaker aperture. In the original WPC plans, the opening was roughly centered in the floor, but my plans in Chapter 21, *Cabinet Body* place it further towards the back, to leave more space for the PC motherboard. From what I've been able to learn about building speaker enclosures, the placement of the opening has little or no effect on the acoustics, so you can move it further back or further forward if that would be more convenient for your setup.

Wiring: Use ordinary stranded hookup wire, in a fairly sturdy gauge. 18 AWG should be more than adequate. The people who sell speaker wire want you to believe you need extremely thick wire for even tiny speakers, but we're working with fairly low-power amplifiers here; you don't need to go overboard.

See the diagrams in the "Amplifiers" section below for specific wiring plans.

Speaker lights

Some people install LED strips around the perimeter of the backbox speakers. See "Installing speaker LED strips" in Chapter 32, *Original WPC Speaker Panel* for a how-to guide.

Amplifiers

The sounds outputs from a PC motherboard or sound card are "line level" outputs, meaning they have to be connected to an amplifier, which is in turn connected to the speakers. Speakers that are made specifically as "PC speakers" generally have their own built-in amps, meaning you meaning you can just plug them straight into the sound card. But this isn't the case when you're working with a standalone speaker designed for a car or pinball machine: for those, you need a separate amplifier.

So you can either use PC speakers, which have their own challenges, as we mentioned earlier, or you can use component speakers and install an amplifier. Assuming you're going with a separate amplifier, let's look at options.

Power levels (Watts per channel)

You're probably accustomed to stereo/home theater receiver amps with power ratings of 100 Watts per channel and up. That's much larger than the power ratings you're likely to see for the sorts of amps we're considering here, and much larger than you need in a pin cab.

The Watts-per-channel rating can be a bit misleading. Most people take it to be an indicator of the loudness that an amplifier can produce. That's basically true, but not quite in the way we tend to think. We tend see these numeric scales as linear, so we think that 100W is twice as loud as 50W. The relationship between power and loudness is actually logarithmic, so the real situation is that 100W just sounds *incrementally* louder than 50W. The rule of thumb is that you have to roughly double the wattage for the ear to perceive any difference in loudness. Doubling the power is like going from "5" to "6" on the volume dial, not like going from "5" to "10".

My point is that you shouldn't be too alarmed if the amps you're looking at have advertised wattage levels well below what you're accustomed to for home theater systems. By way of comparison, the 1990s Williams pinball machines had a whopping *14 Watts* of power *combined* for the speakers in the backbox, and a

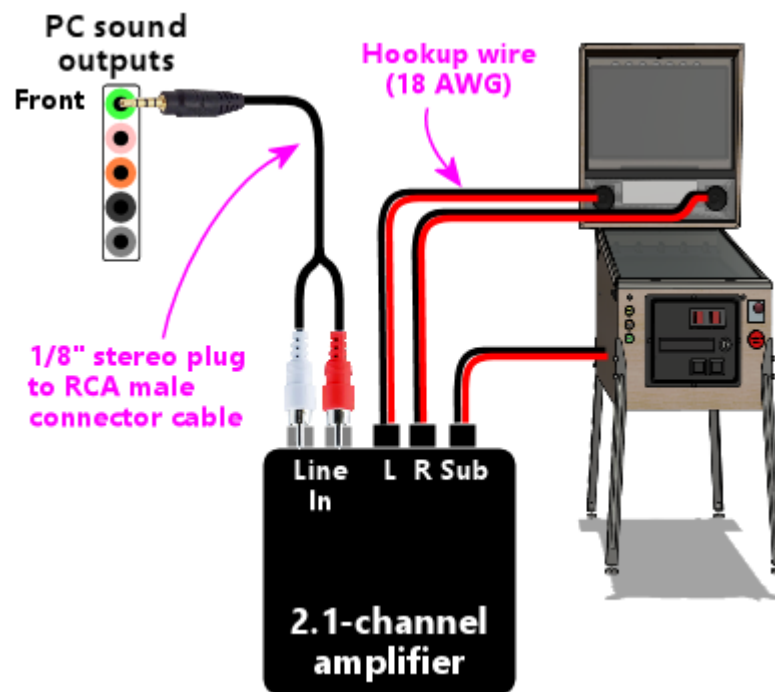
second 14 Watt channel for the subwoofer.

Integrated 2.1-channel amplifiers

For the main speakers, we need three channels of amplification: the left and right backbox speakers, and the subwoofer. The most common way that pin cab builders accomplish this is with a so-called 2.1-channel amplifier - a single unit that has two main channels (that's the "2" in "2.1") and a subwoofer channel (the ".1").

The advantage of an amp designed for 2.1 channels is that it should have a built in "crossover", which is a little filter circuit that sends the higher-frequency part of the signal to the main speakers and the lower-frequency part to the subwoofer. This lets each speaker reproduce the range of frequencies it was designed for, which makes them sound better than if you didn't do the filtering.

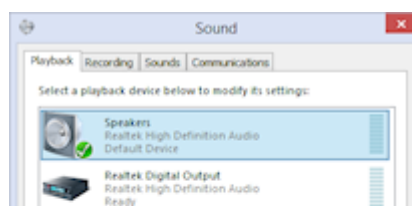
Here's how you wire a 2.1-channel amplifier:



Note: some amplifiers use 1/8" stereo jacks for inputs instead of RCA connectors. Substitute a cable with 1/8" stereo plugs at both ends in that case.

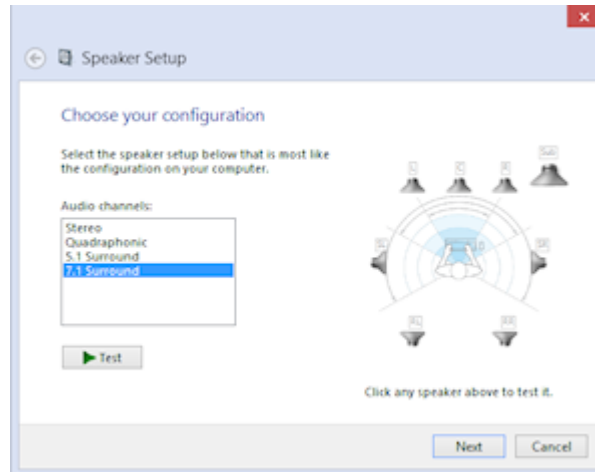
As far as Windows is concerned, there are only two speakers in this setup, even though you have three (left, right, subwoofer). This works because the amplifier has the crossover circuit that divides the signal between the main speakers and the subwoofer. To configure the speakers in Windows, tell Windows that you have "full-range" speakers for the left and right speakers:

- Press Windows+R, type **mmsys.cpl**, press Enter
- Select the Playback tab
- Select your speakers from the list
- Click **Configure**

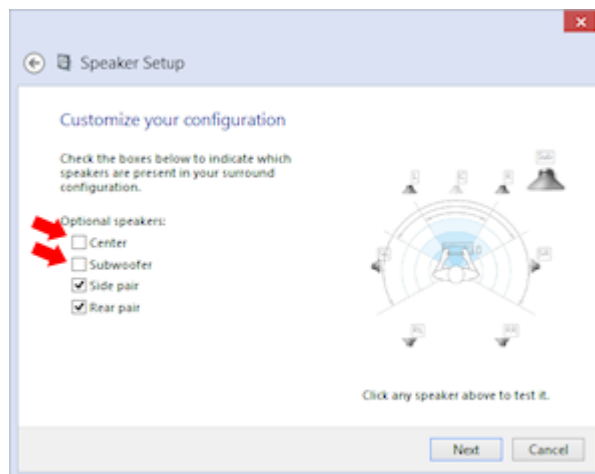




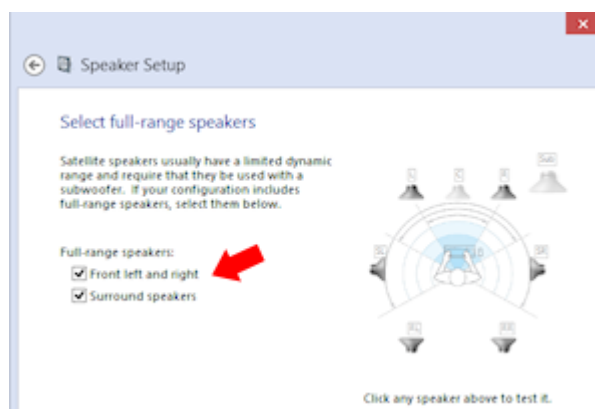
- Select **7.1 surround** (even if you're not actually using surround speakers). Click Next.



- **Un-check** Subwoofer and Center. Check-mark Side Pair and Rear Pair if you're using playfield effects speakers, un-check them if not. Click Next.



- Make sure **Front left and right** is checked for Full-range speakers



- Click Next then click Finish

Separate channel amplifiers

2.1-channel amplifiers are convenient, but there are many more options available if you look at single-channel, 2-channel, or 4-channel amplifiers, without the integrated crossover.

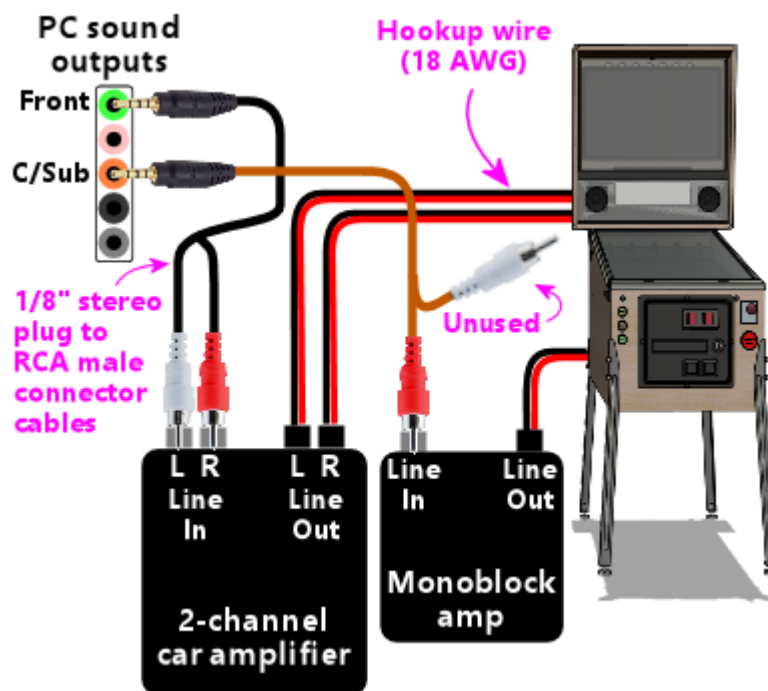
In car audio, the most common setup is to use a 2-channel amplifier for the main stereo speakers and a separate "monoblock" (single-channel) amplifier for the subwoofer. As a result, if you shop for car amplifiers, you'll find tons of 2-channel and monoblock options, and very few 2.1-channel options.

This type of setup is actually easy to implement on Windows, as long as your motherboard or sound card supports 5.1 or 7.1 channel output. The secret is to **let Windows handle the crossover**, so that you don't need a separate crossover circuit in the amplifier, which a 2.1-channel amp would normally provide. Your PC audio outputs should include a jack with "Center/Subwoofer" output, usually color-coded orange.

Why would you want to do this? Because it gives you more options when shopping for amplifiers. You don't have to limit yourself to the small number of 2.1 amplifiers available; you can use just about any car amp.

Wiring with a monoblock amp

Here's how you'd wire a 2-channel amp and a separate subwoofer monoblock amp:

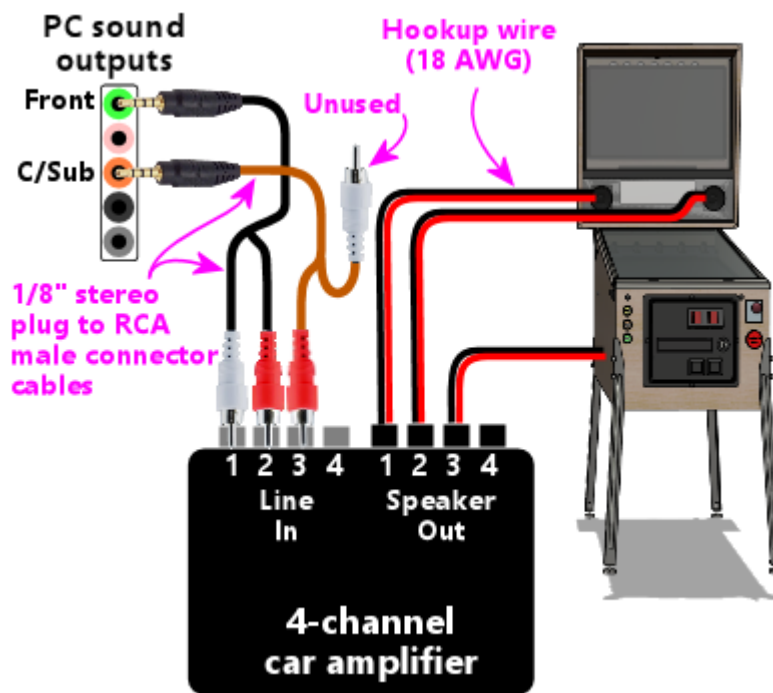


Note: some amplifiers use 1/8" stereo jacks for their inputs instead of RCA connectors. Substitute a cable with 1/8" stereo plugs at both ends in that case.

Be sure to read the section below on configuring Windows for separate subwoofer output. You have to make some settings changes in Windows before it'll send any sound output to the subwoofer jack.

Wiring with a 4-channel amp

Here's how you'd wire a 4-channel car amplifier:



Notes on the separate amplifier plans

Some important notes on both of these plans:

- The Center/Sub output on the PC is a combined output for the center channel and the subwoofer channel. When you plug in a mini-plug-to-RCA cable as shown, one of the RCA jacks will be wired to the subwoofer channel, and the other will be wired to the center channel. None of the current pinball software makes any use of the center channel (it's there for home theater setups, not pinball), so most pin cabs don't connect it to a speaker. That means we have to leave one of the RCA plugs on this cable - the one for the center channel - unplugged.
- There's unfortunately no rule about how the Center/Sub output is wired to the RCA plugs. The Center might be the white plug and the Subwoofer might be the red plug, or it might be the other way around. I don't think there's any way to find out other than trial and error, so if you can't get any sound to come from the subwoofer using the red RCA plug, try the white RCA plug instead.
- Be sure to protect the unused RCA plug on the center/sub cable so that it doesn't accidentally come into contact with anything. It's still electrically connected to the sound card so you don't want it touching a power connector or anything else. Cover the metal end with electrician's tape, perhaps.
- Some four-channel car amps let you "bridge" channels 3 and 4 to create one subwoofer channel with twice the power. Bridging is only possible if the amplifier is designed for it, and the exact method to enable it varies. Check your amp's instruction manual to find out if bridging is possible at all, and how to enable it if so.

Follow the instructions below to configure Windows for this setup.

Windows configuration for separate subwoofer output

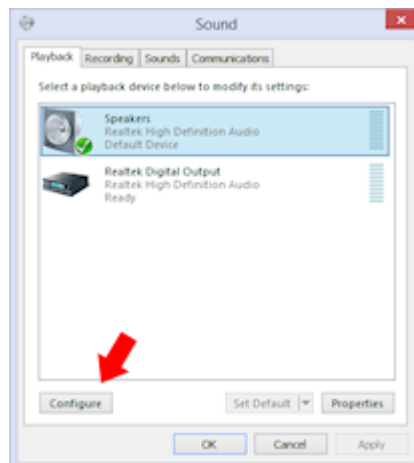
If you're using the subwoofer output from your PC audio output, you have to go through some extra steps to make Windows handle the crossover, so that Windows

distributes the sound properly between the main speaker and the subwoofers.

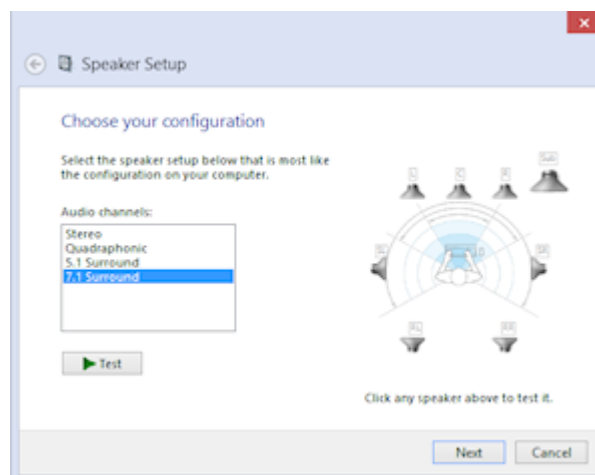
In particular, you have to tell Windows that your front speakers are **not** full-range speakers. "Not full-range" means that they shouldn't receive the low-frequency part of the signal. You also have to enable "Bass Management", which tells Windows to send the subwoofer that low-frequency portion of the signal that it's *not* sending to the main front speakers.

Here's the procedure:

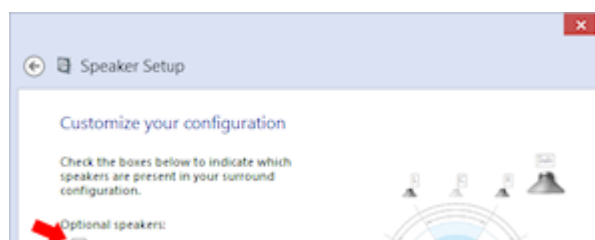
- Press Windows+R, type **mmsys.cpl**, press Enter
- Select the Playback tab
- Select your speakers from the list
- Click **Configure**



- Select **7.1 surround** (even if you're not actually using surround speakers). Click Next.

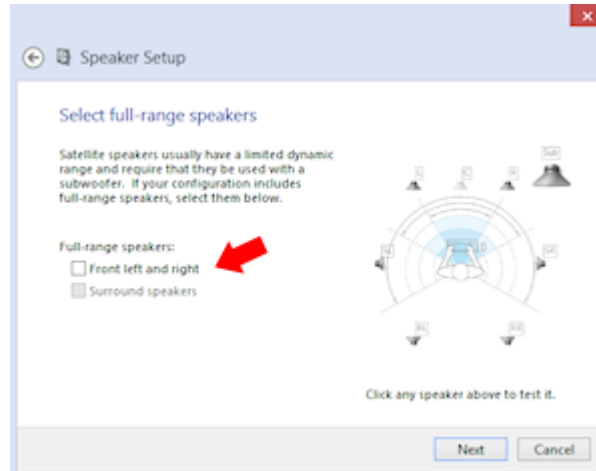


- **Check** Subwoofer, **Un-check** Center. Check-mark Side Pair and Rear Pair if you're using playfield effects speakers, un-check them if not. Click Next.

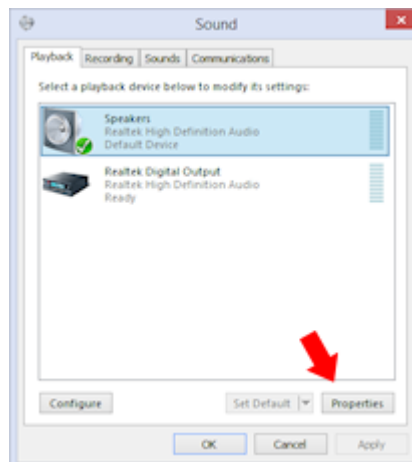




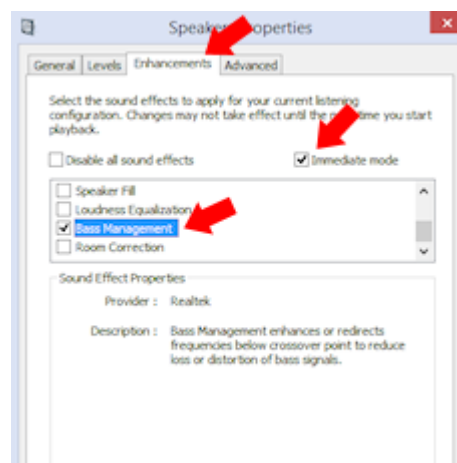
- **Un-check** Front left and right in the Full-range speakers list

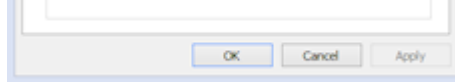


- Click Next then click Finish
- Back in the playback device list, click on the **Properties** button



- Go to the **Enhancements** tab





- Check-mark **Immediate Mode**
- Check-mark **Bass Management** in the list
- Click OK

Car amplifiers

Audio amplifiers for cars run on 12VDC (like almost everything else in a car), which makes them easy to adapt to a pin cab, where we already tend to have a 12V power supply handy. And there's a huge selection of car amps available.

The biggest reason to consider car amps is that they tend to have reliably good audio quality, especially compared to the other options we'll see below. I ended up using a 4-channel car amplifier for my main speakers (the backbox and subwoofer channels) after trying the Lepai and DIY amplifier types and deciding that their sound quality was unacceptable.

There are three main downsides to car amps:

- They're relatively expensive. Expect to pay at \$100 for a decent one, and upwards from there.
- They're big. Many are too big to fit comfortably in a standard cab. Check the dimensions before buying anything to make sure there's room.
- It's difficult to find 2.1-channel options. Most are either 2-channel or 4-channel. However, this isn't necessarily a problem, as you can use the "Separate channel amplifiers" setup described earlier.

DIY 2.1 amplifiers

There are lots of inexpensive amplifier boards available these days designed for hobbyists building Arduino projects or DIY audio systems. I call these DIY amplifiers, but they're not DIY in the sense that you have to build them, just in the sense that they're *for* DIYers. The ones I'm talking about are actually fully assembled circuit boards. What makes them DIY is that they don't come with enclosures; they're just bare circuit boards. Which works fine in a pin cab, where the cabinet can serve as the enclosure, just like for the PC motherboard.

You can find many options from Chinese sellers on eBay, and several are available on Amazon.

I've had poor results with these so far, unfortunately, so I don't have any first-hand recommendations to offer. The ones I've tried had unacceptable background noise levels (that is, noise playing through the speakers when no audio was playing on the PC). This problem actually afflicts a lot of amplifiers in a pin cab environment, because we power them with switching power supplies. An amplifier needs quite good power line filtering to sound good with a noisy power source, and most of these hobbyist boards have little or no power conditioning, to keep the cost down.

All of the DIY amplifier boards are based on specific integrated circuit chips that do most of the amplifier work, so when you go shopping for these, you'll find them identified primarily by the type of IC chip they use. These are all no-brand products, though, so one board based on a particular chip might be great, and another board based on the same chip might be terrible. It's a crap shoot if you buy these on eBay. I think your best bet might be to buy these on Amazon, where you can at least

compare user reviews, even if those aren't perfectly reliable.

Some people on the forums have reported good results with **TPA3116D2-based boards**. I've actually tried one of these and found it to be unacceptably noisy at low input signal levels, although it did well at playing loud sounds. But the chip itself seems to be very well regarded among audio hobbyists, and there are some newer TPA3116D2-based boards available on Amazon that get good user reviews, so one of these might be worth a try despite my experience.

Be sure to look for a board that works on an ordinary DC power supply. Some of the older amplifier IC chips needed AC power supplies (using transformers) or require unusual DC voltage levels. The newer chips are mostly designed for more common DC voltages like 12V or 24V.

Advantages:

- Inexpensive - \$25 to \$40
- Available in 2.1 configurations with crossover
- Compact, easy to fit in a cab
- Relatively high power levels are available (TPA3116D2 amps are nominally 50W/channel with the right combination of power supply and speakers)

Disadvantages:

- No-brand products, so quality is hit-or-miss

Lepai LP-168HA 2.1 amplifiers

The LP-168HA is a 2.1-channel amp, which is what makes it popular among pin cab builders. It's so difficult to find 2.1-channel options that I think this one became popular by default. VirtuaPin used to sell these as part of their speaker packages - they don't seem to offer them any more, but you can easily find them on Amazon and eBay.

This has long been the go-to amp for most pin cab builders. I'm afraid I haven't had good experiences with it, though. I've tried two of them; the first one performed so badly that I assumed it was defective, so I sent it back for a replacement, and that was just as bad.

The problems I had with the Lepai were all with its audio quality. It had a lot of background noise with no signal playing; it was too underpowered to produce even modest volume levels with my 4" backbox speakers; and the crossover basically didn't work (on either unit I tried), making it almost impossible to get a proper volume balance on the subwoofer - the sub would be either off or driven to total distortion, with nothing in between. I've seen reports of the same problems on the forums and Amazon reviews, so I tend to think these reflect design flaws, but they could merely be common defects that only affect some percentage of units.

To be fair, some people on the forums have said they're happy with these amps, so I might have just had the bad luck to get two particularly bad units.

If want to try the Lepai, be aware that there are a number of identical looking units sold under very slightly different names, like "Lepy" and "Lepei". I don't know if Lepai just can't decide how its name ought to be rendered in a Western alphabet, or if the variations are knockoffs (I'd say "cheap knockoffs", but the original was already cheap). Maybe it's a mix of both. For what it's worth, I've talked to a couple of people using the maybe-knockoff brands who were happier with the results than I was with the (I assume) original brand.

Other packaged 2.1 amplifiers

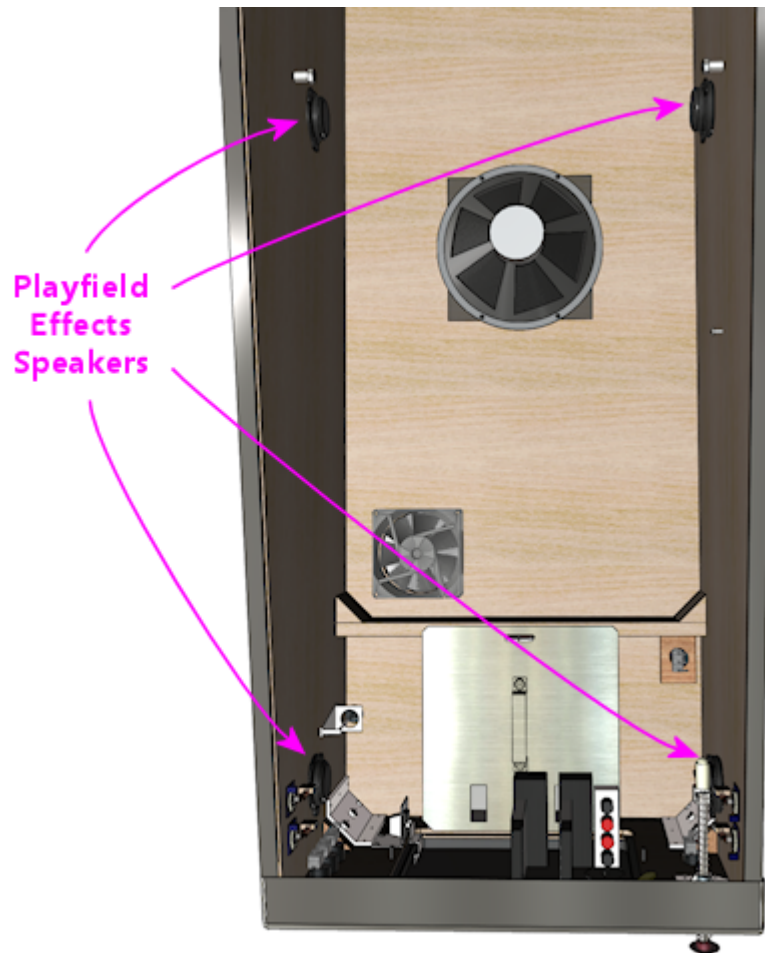
There are a few other packaged 2.1-channel amplifiers, similar to the Lepai above, available on Amazon and eBay. (By "packaged", I mean that they come in enclosed cases, not just raw circuit boards like the DIY amps mentioned earlier.) Some of them are newer designs based on more powerful chips like the TPA3116D2, and I wouldn't be at all surprised if at least a few of these are just repackaged versions of the DIY boards. I haven't tested any of these myself.

Home stereo receivers

I don't know of anyone who's done this, but a home stereo receiver could potentially be re-purposed as a pin cab amplifier. These tend to have excellent sound quality, even the cheaper ones. The challenge is that they tend to be much bigger than you could comfortably fit in a pin cab.

Playfield effects speakers

In addition to the main backbox speakers, a separate set of speakers can be placed inside the cabinet, usually under the TV where they can't be seen, to reproduce "mechanical" sound effects - the sounds made by things on the playfield, like the ball rolling around and bumping into things, flippers flipping, bumpers bumping, and so on.



Newer versions of Visual Pinball have support for a "surround sound" system for the playfield effects. This isn't quite the same as a home theater surround sound setup, where you'd place speakers at the sides and back of the room to create a 360° sound field that surrounds the listener on all sides. For a pin cab, we borrow the same multi-channel technology they use in home theaters, but instead of using it to

surround the listener, we use it to make the sound effects sound like they're coming from specific points on the playfield. So the thing we're "surrounding" is the playfield, not the listener. As such, we place the speakers as shown above, at the corners of the playfield area inside the cabinet.

Older versions of Visual Pinball (before 2017, when the surround feature was added) had a more primitive version of the feature that allowed you to play the mechanical effects through speakers in the cab, but without the multi-channel capability. It at least created the illusion that sounds were coming from the playfield area, but only generally, since sounds couldn't be positioned in space the way they can with four speakers.

Equipment

To set up a surround-sound playfield effects system, you need:

- Four speakers
- Two 2-channel or 2.1-channel amplifiers

Speakers/excitors

The playfield effects are just another set of audio channels, so at a basic level, you just need another set of four speakers.

The best type of speaker for this job seems to be something called an "exciter", also known as a tactile transducer or tactile subwoofer. An exciter is like a speaker without the paper cone part. They're designed to be attached to a rigid surface, and they work by making that attached surface vibrate. The surface takes the place of the paper cone in a normal speaker. In our case, the wall of the cabinet serves as the surface.

One reason that exciters work well for this job is that they're smaller than regular speakers. It's a lot easier to find space for them in a cab. And they're designed to mount to a flat, rigid surface, which is a perfect fit here, since we can use the side walls.

The other reason they're so good for this job is that they're specifically designed to produce a tactile effect for low-frequency sounds, which is precisely what we want from the playfield effects. The playfield effects are all meant to simulate mechanical things on the playfield moving and around and bumping into each other.

There are many options for exciters and tactile subwoofers available online from Amazon and other Web sellers. I think any exciter that gets decent user reviews on Amazon would be fine, since this isn't exactly the most demanding audiophile scenario; these speakers are mostly for percussion-type effects, not music or voices. So I'd recommend doing a little research on Amazon to see what's currently on offer. At the risk of listing equipment that may no longer be available by the time you read this, here are some specific exciters that forum members have mentioned favorably:

- Dayton DAEX25
- Dayton Audio DAEX25VT-4
- Dayton Audio DAEX58FP

You can use regular speakers if you prefer, but I don't think there are any advantages. Regular speakers are larger and less tactile.

In the days before VP's surround sound support, some people set up one- or two-speaker systems using their TV's built-in speakers. I don't recommend this approach. Flat-panel TV speakers are invariably small and tinny. They won't reproduce

percussion-type effects with any fidelity.

You should use four identical speakers or exciters for the effects speakers. This helps with the illusion of spatial positioning by matching the tonal quality at each speaker as closely as possible.

Amplifiers

For four speakers, you need four amplifier channels. This is in addition to the amplifier(s) you're already using for the main backbox speakers and subwoofer.

The usual setup is to add two more 2-channel or 2.1-channel amplifiers. Use one for the front pair of exciters, and the other for the rear pair.

I'd recommend using one of the DIY amplifier boards mentioned earlier, as they're inexpensive and compact, and the ones based on newer chips like the TPA3116D2 produce decent sound quality. I'm personally a lot less picky about audio quality for these amps than for the main backbox speaker amp, since these speakers are mostly for percussion-type sound effects, not for music or voice effects.

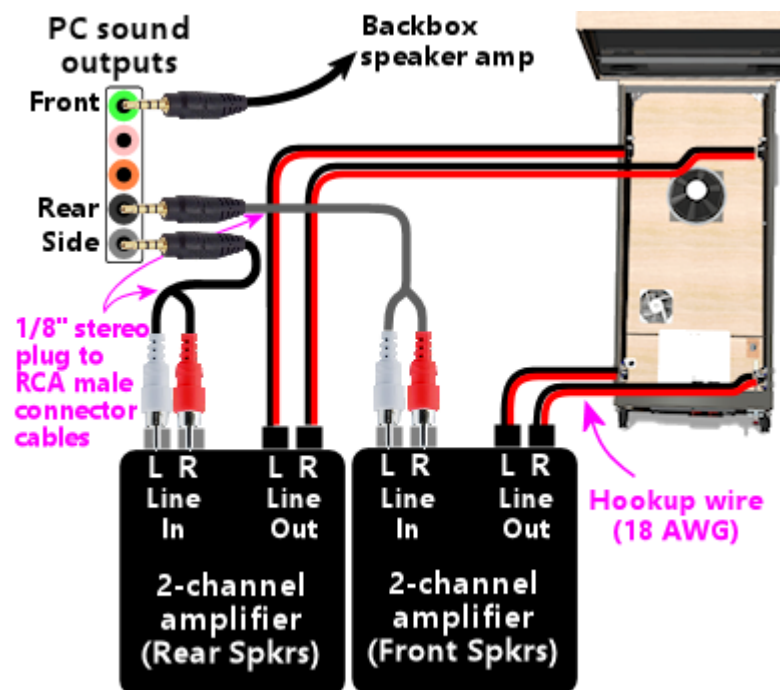
Where to install the playfield effects speakers

For a four-speaker surround system, the speakers should go roughly at the corners of the playfield TV.

Exciters are designed to mount on flat surfaces. The side walls of the cabinet are perfect for this. I'd mount the exciters on the side walls just below the TV, being sure to leave enough vertical clearance for the TV.

Wiring the playfield effects speakers

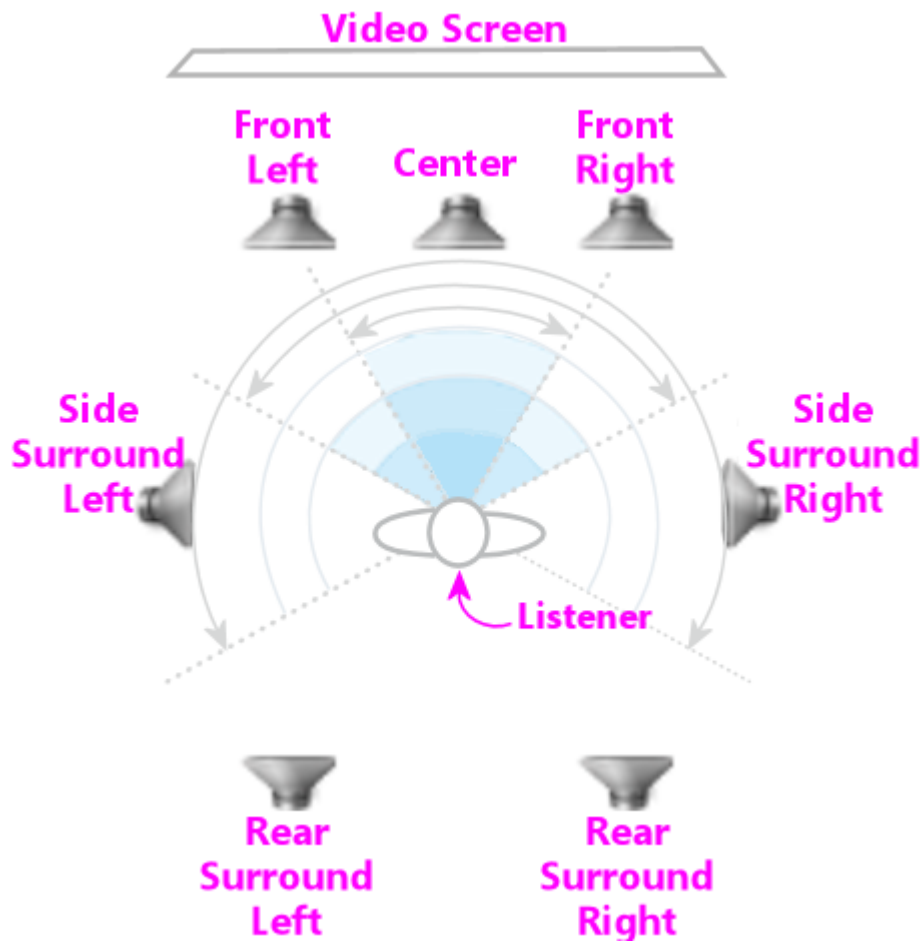
The wiring plan for the playfield speakers is very similar to the plan for the backbox speakers. The main difference is that we need two of the 2-channel amps now, since we have four speakers on four separate audio channels. It's most convenient to think of these as two pairs of stereo speakers - a stereo pair at the front and a stereo pair at the back. Each stereo pair connects to one of the amps, using the normal Left/Right stereo hookups on the amps.



Key features to note:

- The **Front** output jack on the PC remains connected to the main backbox speaker amplifier as before - make no changes to that
- Use **two** 2-channel amplifiers, one for the front left/right speakers, and one for the rear left/right speakers
- The amplifier for the **front speakers** connects to the **Rear Surround** audio jack on the PC
- Let me say that again, because it's too crazy to read right the first time: the **Front** speakers plug into the **Rear Surround** jack
- The amp for the **rear speakers** connects to the **Side Surround** audio jack on the PC

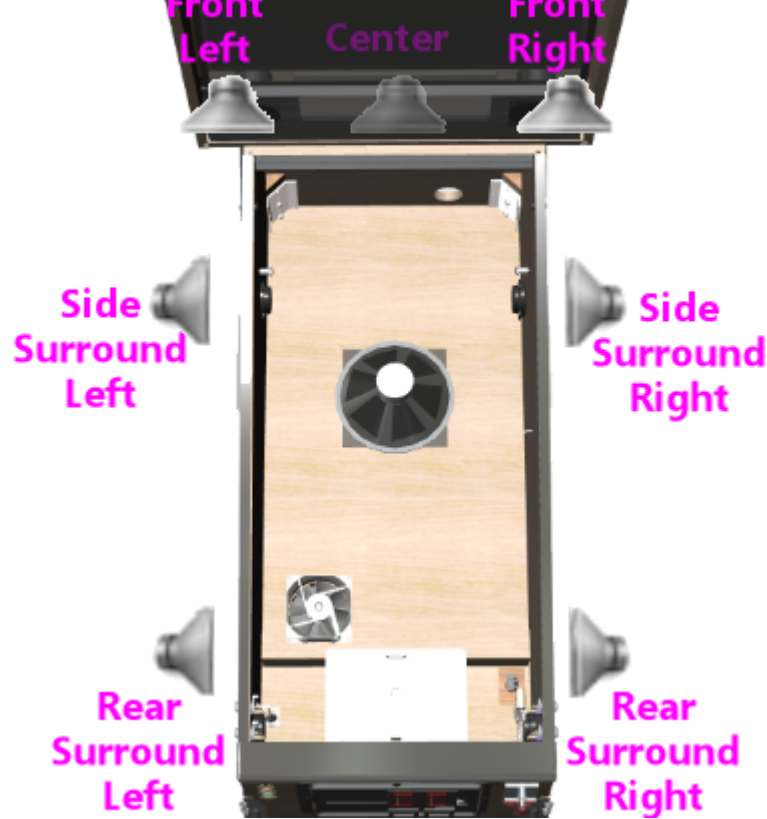
What's up with that bizarre wiring with the **Front** speakers connected to the **Rear** surround jack on the PC? I know it sounds crazy. The way to make sense of it is to think about the way surround sound works in a **home theater** setup. The surround sound feature in Windows is all designed around the home theater way of thinking. Home theater people think in terms of a speaker layout like this, with the listener at the center, and speakers placed around the perimeter of the room:



This is how Windows sees the 7.1 audio format. The format is designed with home theaters in mind, so it assumes this particular spatial layout. This is an overhead view; the figure at the center is the listener.

Windows is very attached to the idea that the speakers have this specific spatial layout. When the Visual Pinball developers were adding the surround sound feature, they had to work with that layout. So how does this map onto a pin cab most easily? Like this:





So hopefully the twisted logic becomes more apparent now:

- Windows "Front Left" and "Front Right" = the main backbox speakers
- Windows "Center" = unused
- Windows "Side" = **rear** cabinet speakers (towards the back of the cab)
- Windows "Rear" = **front** cabinet speakers

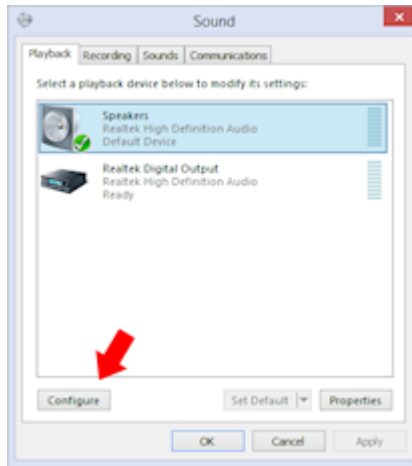
Now we can see how we got to that confusing last element, where what Windows calls "Rear" corresponds to what we think of as the **front** of the cabinet. Remember that Windows thinks about this in home theater terms, where the listener is in the middle of the picture, rather than standing at one end. You have to picture the listener sitting somewhere in the middle of the playfield for Windows's idea of "Side" and "Rear" to make sense.

Also note that the "Front Center" speaker in the Windows layout isn't used at all. We don't even connect a physical speaker there. If this speaker were present, it would have to be situated right in the middle of the speaker panel. We can't put a speaker there because that's where the DMD (score display) goes. Even if we could fit a speaker there, there wouldn't be any benefit sonically, since the left and right speakers are so close together. The center channel in the 7.1 audio format is intended for home theater systems, where the front left/right speakers might be placed six or eight feet apart. In that case, the sound field is so wide that it's helpful to have an extra speaker in the middle, to keep the dialog sounding like it's coming directly from the screen. That extra degree of localization is pointless in a pinball setup, since the left and right speakers are so close together that the ear can't really localize sound to one or the other anyway.

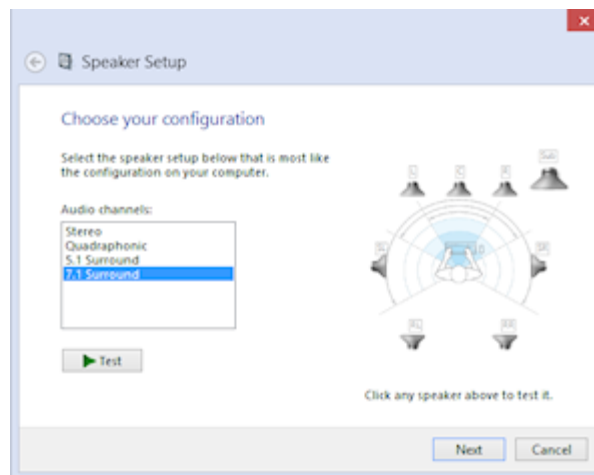
Configuring Windows for playfield effects speakers

- Press Windows+R, type **mmsys.cpl**, press Enter
- Select the Playback tab
- Select your speakers from the list

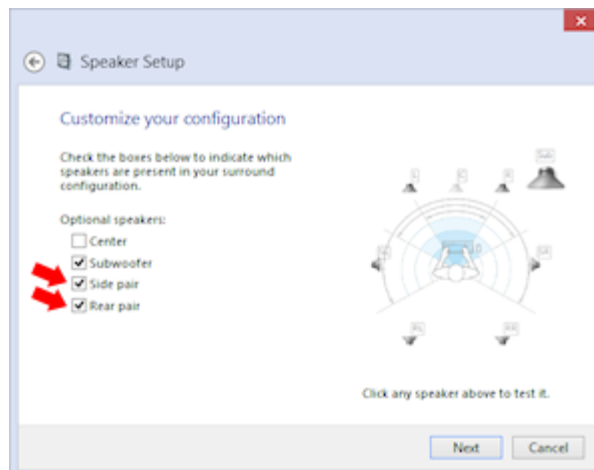
- Click **Configure**



- Select **7.1 surround**. Click Next.



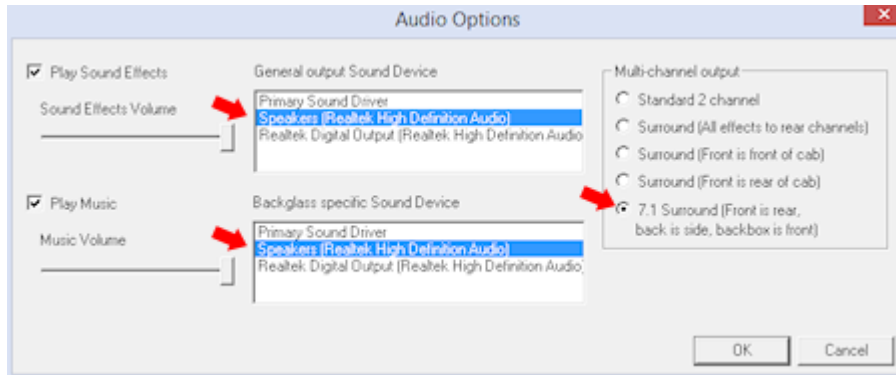
- **Check** Side Pair and Rear Pair. **Un-check** Center. Leave Subwoofer as before, according to how you set it up for the main backbox speakers. Click Next.



- Leave the "full-range" settings as before, according to how you set it up for the main backbox speakers. Click next.
- Click Next then click Finish

Configuring Visual Pinball for playfield effects speakers

- Launch Visual Pinball, without loading any game yet
- On the menu, select **Preferences > Audio Options**
- In the **General output sound device**, select your main sound card/speakers from the list. It's better to select the device specifically rather than the default "Primary Sound Driver", since that doesn't always work.
- In the **Backglass specific Sound Device**, select the same device
- In the **Multi-channel output** section, select **7.1 Surround**



If that little homage to Orwell saying "Front is rear, black is white, war is peace" that you see in the parentheses after "7.1 Surround" seems confusing, it's because VP is trying to explain the whole speaker layout in 10 words or less to fit the dialog box. Here's what it's trying to say:

- Your pin cab's main backbox speakers connect to the PC audio "Front" output jack
- Rear playfield effects speakers connect to the PC "Side Surround" output jack
- Front playfield effects speakers connect to the PC "Rear Surround" output jack

This can seem backwards at first glance, but it makes a kind of sense when you take into account how Windows thinks about surround sound. The section above on wiring the effects speakers has a more detailed explanation of the Windows surround sound model and why the connections have to be arranged like this.

Editing Visual Pinball games to send sounds to the backbox speakers

If you have playfield effects speakers set up and configured in Visual Pinball, VP's rule for deciding when to use which speakers is really simple:

- If the sound comes from the game's ROM (the original game's software, being emulated in VPinMAME), it's played through the backbox speakers
- Otherwise, it's played through the playfield effects speakers

That rule usually does exactly what you want, because almost all of the sound effects that aren't from the ROM are meant to simulate something mechanical on the playfield. In some cases, though, you might prefer for some of the non-ROM sounds to be played through the backbox speakers. This might be desirable, for example, if you're adding your own extra music or voice effects to supplement the game's original soundtrack. It might also be better for certain mechanical effects, such as EM-era bells (which were often situated in the backbox in the originals) or scoring reel sounds.

VP lets you override the rule on an effect-by-effect basis, so that you can redirect specific sound effects to the backbox speakers. See "How to play table sound effects

through the backbox speakers" in Chapter 18, Customizing VP Tables.

Using playfield effects speakers instead of feedback devices

"Poor Man's DOF" or "Surround Sound Feedback" (PMD or SSF) refers to using playfield speaker effects to replace all of the tactile feedback effects that many cab builders implement with DOF using contactors, solenoids, and the like. The main difference between this and the basic playfield speaker setup is that some PMD/SSF builders add extra exciters to strengthen the tactile effect, particularly at the front of the machine where it's more noticeable. For example, some people put an exciter under the lockbar, since that's where you rest your hands while playing.

For more information, see the SSF group on Facebook:

www.facebook.com/groups/SSFeedback/

I personally prefer discrete feedback devices for the solenoid effects, as I find their audible and tactile effects more convincing than audio recordings. I see playfield effects speakers as a great complement to DOF, for other non-solenoid noises such as the ball rolling and colliding with things. But the PMD/SSF approach is attractive to some people for its lower cost and lower complexity.

Volume controls

Your amplifiers probably have volume knobs. But here's the problem: do you think you're going to want to open up your cabinet and adjust those knobs every time you want to turn the sound up or down? Certainly not. You're going to want some kind of external volume controls instead.

Pin cab builders over the years have come up with several ways to approach this. Some of the early cab builders were stuck on the idea that you had to use the volume knob to adjust loudness, so they came up with ways to accomplish that without having to take apart the cab every time:

- Situate the amplifier near the coin door, so that you can reach in through the door and turn the knob
- Install the amplifier so that the knob actually sticks out through a hole in the side of the cabinet, so that you can turn the knob without even opening the door
- Install a remote-controlled motor that turns the knob for you when you push buttons on the remote

My advice is to stop fixating on the volume knob, and use a whole different approach: **let Windows control the volume**. Windows has its own notion of the line output volume, which can be adjusted in software. Doing it software means that you can control the volume with the keyboard or mouse. That greatly simplifies the physical controls, because you no longer have to worry about how to reach the volume knob on the amplifier.

Pre-set the volume knob

To let Windows control the volume, the first step is to set a **fixed reference level** for the volume controls on your amplifiers. You'll turn the amplifier knob to this setting, and then *just leave it there from that point on*. When you want to adjust how loud a game sounds, you *won't* open up the cab and turn the knob. You'll change the Windows volume level instead.

It's important to understand that the function of a volume knob on an amplifier is

turn **down** the power. An amplifier has an intrinsic maximum power level, which is a function of the way it's designed. If you didn't have the volume knob at all, the amplifier would simply run at that maximum power level. The volume knob's function is to reduce the power level from that maximum to whatever lower level sounds right to you. When the volume knob is turned all the way up, it means that you're letting the amp run at full power - you're not attenuating the power at all.

So in principle, the fixed reference level for any amplifier should simply be what you get when you turn the volume knob all the way up.

In practice, though, you usually don't want to do that. The problem is that an amplifier amplifies not only the audio signal but also the random background noise that's always present on the signal input. When you turn the knob all the way to 10, the amplification is usually so strong that it exaggerates the background noise, so that you hear a constant loud hiss or buzz when there's no audio input signal playing.

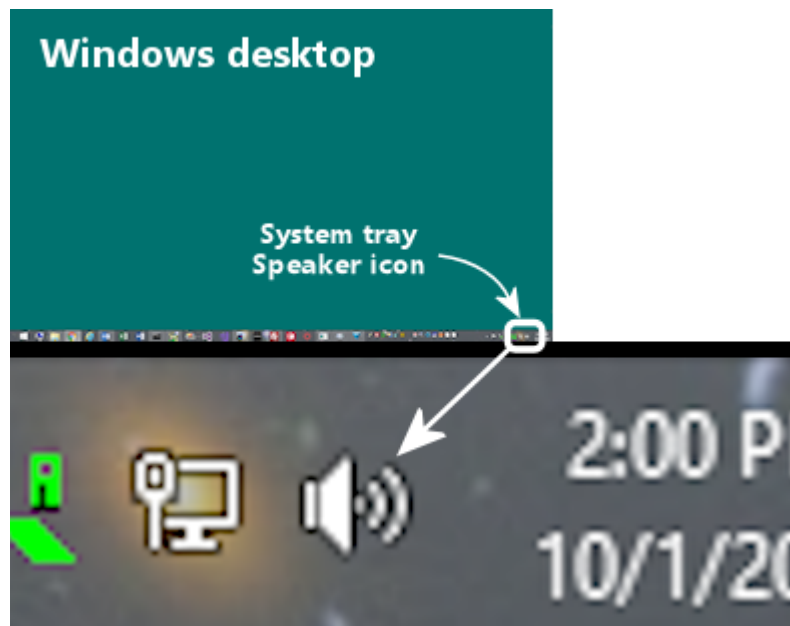
So what you want to do is find a reference level that's as high as possible, without producing excess hiss or buzz when the audio input is quiet.

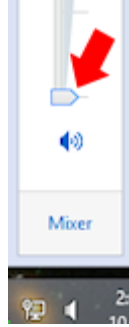
The procedure to find this level is pretty simple. Get everything connected and turn on the amplifier. Make sure it's connected to the PC output jack so that this is a fair test of normal playing conditions, but make sure Windows isn't playing any sounds. Turn the volume knob on the amp all the way down. Now turn it up slowly. Keep going as long as the background noise coming out of the speakers doesn't get excessive.

If you have a really good amplifier and good wiring, you might be able to turn the knob all the way up, or very close, without excess noise.

To test that this setting is loud enough for normal use:

- Turn the Windows master volume level all the way down, using the volume controls in the "system tray" at the bottom of the screen





- Play some sample music in your a media player, just to test the level
- Gradually turn up the Windows master volume until it's as loud as the loudest level you'll likely want to use for pinball simulations
- If that's less than 100% on the Windows volume knob, you're done
- If you get to 100%, and it's still not loud enough, turn up the volume on the physical volume knob on your amp until it's loud enough
- Turn the music off and re-check the background noise level coming from the speakers. If it's acceptable, you're done.
- If there's too much noise at the new physical volume knob setting, try turning the physical volume knob back down until the noise is okay. Then repeat the music test.
- You might have to repeat the loud/quiet test a few times to find the optimal balance between "loud enough" and "quiet enough". Some of the cheapie 2.1 amplifiers and DIY amps can be pretty noisy when turned up to high volumes, so you might have to put up with a certain amount of background noise to get enough loudness at the high end. Or, conversely, you might have to accept a limit on maximum loudness to make it quiet enough.

PinVol

I wrote a little utility program called PinVol that helps with audio volume management, specifically for pin cabs. It's free and open-source. Find out more about it here:

mjrnet.org/pinscape/PinVol.html

PinVol lets you assign any keyboard keys or joystick buttons to serve as volume controls. It also has the notion of a "global" volume level and a separate "local" volume level for each table, which is designed to help you equalize the the loudness level across different tables. Some VP tables are much louder than others. PinVol remembers the per-table volume setting for each table and automatically restores it each time you run a table, so that you don't have to keep manually changing the volume level every time you switch tables (which I found myself doing constantly, because of the big variations in loudness from one table to the next).

With PinVol, you can assign different keys to control different aspects of the volume:

- "Global" volume keys to control the system-wide volume
- "Local" volume keys to control just the volume level for the current table
- Global mute, to silence all audio effects

Set up physical buttons for controlling the volume

The next piece of the puzzle for controlling the volume through Windows is to set up some physical controls to adjust the Windows master volume.

If you're not using PinVol, you typically just need two or three buttons: Volume Up, Volume Down, and Mute.

If you're using PinVol, you'll want at least four buttons: Global Volume Up, Global Volume Down, Local Volume Up, and Local Volume Down. You might also want a Mute button and/or a Night Mode button.

There are several common options for setting up physical buttons:

- If you're using the Pinscape Controller or an i-Pac as your key encoder, you can use "shifted" buttons for the volume controls. Shifted buttons let you assign two separate functions to each physical button - a normal function and a "shifted" function. The shifted function is engaged by holding down another button - the Shift button - and pressing the first button.

For example, I use the Extra Ball button as my Shift button, and I use the shifted flipper and MagnaSave buttons as my volume controls. The flipper buttons are just flipper buttons most of the time, but when I hold down the Extra Ball button, my right MagnaSave/Flipper buttons become the Table Volume Up/Down buttons, and the left ones become the Global Volume Up/Down buttons. I find that pairing the buttons on each side as an Up/Down pair is intuitive and easy to use.

This is my favorite approach because it's so convenient and it doesn't require any additional physical controls. To set this up in the Pinscape Config Tool, go to the button assignment section, and read the on-screen instructions for setting up a Shift button.

- Add a rotary encoder dial. This can be mounted anywhere a button can be mounted, but it gives you a combined Up/Down control in one small knob, so it's somewhat less conspicuous than a pair or trio of buttons. Some dials can also act as a pushbutton when you press the knob, which makes an intuitive place for the Mute button, giving you three controls in one. See this thread on vpfforums:

www.vpfforums.org/index.php?showtopic=42812

This is my second-favorite option after using "Shifted" buttons. A lot of people consider it their top choice because a knob is so natural as a volume control. The only reason I rank it second-best is that it *is*, after all, another control.

- Add some more front-panel pushbuttons, of the same type as the Start and Exit buttons. Most people don't like doing this because of the excess clutter, but you can mitigate the clutter by using smaller buttons or small rocker switches, and you might be able to hide them somewhat by installing them in the coin door, which is pretty good at hiding things because it's matte black.
- Add pushbuttons or small rocker switches on the bottom of the cabinet. This is nicely hidden, but it's also less convenient to access.
- Add controls inside the coin door. This is also nicely hidden, but it's even less convenient to access than bottom controls.

Software setup for volume controls

The last step in setting up software volume control is to map the physical buttons on your cabinet that you've designated as the volume controls so that they trigger the Windows master volume adjustments.

If you're not using PinVol, the easy way to do this is to assign buttons to the special keyboard keys **Media Volume Up**, **Media Volume Down**, and optionally

Media Mute. These are standard keys on a USB keyboard, and your key encoder will hopefully include them among the keys you can assign to buttons. Windows automatically recognizes these keys and uses them to adjust the master volume control in the system tray, so you don't have to do anything special in your Windows setup - these keys should just work automatically as soon as you assign them to buttons.

If you're using the Pinscape Controller as your key encoder, you can find these keys here on the little mini-keyboard that pops up when you assign keys in the Config Tool:



For other key encoders, look for similar icons, or look for the key names Volume Up, Volume Down, and Mute.

If you're using PinVol, you can assign any keys or joystick buttons as the volume controls. PinVol shows instructions in its main window for assigning the desired keys. Just follow the on-screen prompts.

If you're using a Pinscape Controller for button input, I recommend assigning high-numbered "F" keys, like F14 through F20, for the PinVol hot keys. The PinVol hot keys are global to the entire Windows system, which means that once they're assigned to PinVol, other applications won't be able to use them. The high-numbered "F" keys are a good choice for this because I've never seen any applications use them as default key mappings, so they shouldn't conflict with anything else you're running. Here's the procedure to map them:

- Run the Pinscape Config Tool
- Go to the Settings screen
- Scroll down to the button assignments section
- Assign each button input for a volume control button to the desired keyboard key or joystick button
- Save settings and exit the config tool
- Run PinVol
- Click in one of the key assignment boxes ("Global Volume Up", etc)
- Press the button you want to assign to that function
- Repeat for each button

42. Backbox Toppers

Toppers are extra decorations on top of the backbox, such as *The Addams Family's* cloud or *Whirlwind's* fan. A few titles over the years included toppers as original equipment, but not all that many, so they were a great novelty feature that helped a machine stand out from the crowd in an arcade. In more recent times, pinball collectors have become fond of adding after-market toppers to every machine in their collections that didn't already have one from the factory.

Virtual cab builders like their toppers as well, but often for different reasons. For the real pinball machines, toppers are mostly about novelty and decoration. In the virtual world, in contrast, toppers tend to be more about function than decoration. There are certain feedback devices that just won't fit anywhere else, particularly fans and beacons. We virtual cab builders tend to view the topper area primarily as free space for feedback toys.

I think it's worthwhile in a virtual cab to take both views of the topper - it's a functional feature, but it's also part of the artwork. We all put a lot of effort into our cabs and want them to look as great as they play, so it's worth thinking about how to incorporate the topper elements into our visual themes. It's nice for the topper to feel like a natural extension of the cabinet art, rather than just a bunch of gadgets bolted on top.

Let's start with a look at how some of the real machines used toppers to enhance their visual theming, to get some ideas about how we can do the same thing.

What makes a good topper?

The obvious place to start looking for ideas for a virtual cab topper is in the real machines. If you do a Web search for "pinball topper", you'll find all sorts of after-market toppers for sale. Most of these are add-ons for machines that didn't have factory toppers, though, and for the most part they're not all that imaginative (in my opinion, at least). A lot of them are just acrylic signs that copy some of the cabinet artwork, and most of the rest are some little doo-dad that's vaguely related to the theme.

The best toppers on the real machines, for the most part, are the ones that came as factory equipment. Many of those were thoughtfully integrated into the theme and formed a part of the overall artwork. One of my favorite examples is the taxi cab roof sign from *Taxi* (Williams, 1988). It's the perfect icon for the game's theme, but it also meshes with the backglass art, sitting right on top of the taxi depicted in the art. It's a clever extension of the backglass painting into the 3D space above the backbox.





Along the same lines, a number of police-themed pinballs over the years featured rotating beacons, or even complete police-car light bars, as toppers. Police themes are nearly as common in pinball as in CBS dramas, so there are lots of examples, but the canonical one is *High Speed* (Williams, 1986), where the beacon fits into the backglass art using exactly the same trick as *Taxi*'s roof sign:



A beacon isn't just a passive decoration, either. It can be an active part of the game action, lighting up and spinning in sync with events in the game. It's no wonder they used these over and over on the real machines. They even used them in a few cases where it's hard to see any relation to the theme. One curious example is *F-14 Tomcat* (Williams, 1987), which had not one, not two, but *three* beacons.



I wouldn't consider this one to be good integration of topper and theme. Even so, it's understandable why Williams used beacons so often, since they add not just an extra decoration, but also a functional element, in the form of a light show.

Another great functional topper is the fire bell on *Fire!* (Williams, 1987). It's the sort of bell you'd expect to see on an antique fire wagon, so it's a good theme icon, even if it doesn't meld as seamlessly with the artwork as the *Taxi* sign and *High Speed* beacon. Like the police beacons, it's an active part of the game's sensory effects - it has a solenoid hammer that rings the bell when certain events occur in the game.



Yet another brilliant functional topper is the cloud-shrouded fan on *Whirlwind* (Williams, 1990). It's the perfect active toy for the theme; the fan activates during multiball modes and other events in the game, creating a tactile sensory effect from the blowing air. The cloud shape of the enclosure meshes with the backglass artwork, extending the stormy sky in the art into the space above the backbox.



The somewhat similar cloud topper on *The Addams Family* (Midway, 1992) takes the same approach, and also serves as the top of the Addams mansion, whose roof extends out of the frame at the top of the backglass. It doesn't have anything as

dramatic as the *Whirlwind* fan, but it has flashers inside that simulate a lightning storm when you start a multiball mode.



Funhouse (Williams, 1990) didn't have a topper as original equipment, but here's an example where someone came up with an add-on topper that's as good as anything that came out of the factory. Heinz-Peter Bader designed this topper for his own *Funhouse*. He took a vanity mirror light - the kind with a row of big round bulbs, like in the stereotypical actor's dressing room mirror lights - and filled it out with party bulbs in assorted colors, then put it on top of the machine with the bulbs pointing up. (I've seen at least one other similar *Funhouse* topper mentioned in a pinball forum, so I'm not sure who came up with the idea first, but this is certainly a nice implementation in any case.)



Photo courtesy of Heinz-Peter Bader

It's interesting to compare this design to the various after-market commercial toppers for *Funhouse* that the "mods" companies sell. The bulb topper works so well because it's perfectly in scale and it captures the carnival atmosphere of the theme, plus it's a natural extension of the backglass artwork (which is already festooned with little marquee bulbs) into the space above. In contrast, all of the commercial topper I've seen for *Funhouse* just reiterate something from the artwork, without feeling like part of it: a giant "Admit One" ticket, a cut-out of Rudy. They do nothing to merge with the backglass art.

Virtual cab toppers

Virtual cab builders usually take a utilitarian view of the topper, focusing on the functional elements. There's a fairly standard set of feedback devices that most cab builders incorporate in their toppers:

- Chapter 63, Fan
- Chapter 57, Beacons
- Chapter 56, Flashers
- Chapter 56, Strobes

Most cabs have at least one of these, and many use all of them. They're all great additions functionally. Beacons, flashers, and strobes all make nice light shows, and a fan adds a wonderful tactile element that movie theaters would market as a "4D experience" with a \$10 ticket surcharge.

But does a big pile of devices make for a "theme"? Well, from one perspective, it actually does. One way to see a virtual cab is as a whole arcade's worth of pinball tables in one box. So you could say that the theme is "all the pinballs" - which makes "all the topper toys" an apt fit.

Personally, though, I like the idea of giving a virtual cab some kind of theme more personalized than "all the pinballs". I discuss that idea a bit in Chapter 22, Cabinet Art. If you *are* giving your machine a specific theme, I think it's also a nice touch if you can extend your theme to the toppers.

Of course, functionality is still the starting point in choosing which devices to include. I'd never suggest that you should forego a fan just because your theme doesn't have an obvious connection to air circulation. But maybe you can find ways to work some of the functional elements into your theme, even though the theme isn't the reason you included them. For example, on my cab, I came up with a custom plastic enclosure for the fan that's meant to look like a star-burst, to go along with the outer space theme of my cab graphics. So I didn't really try to make the functionality of the fan fit the theme, but I did at least try to work its shape into the visual layout. (If you're interested in my specific topper design, you can find some more details below.



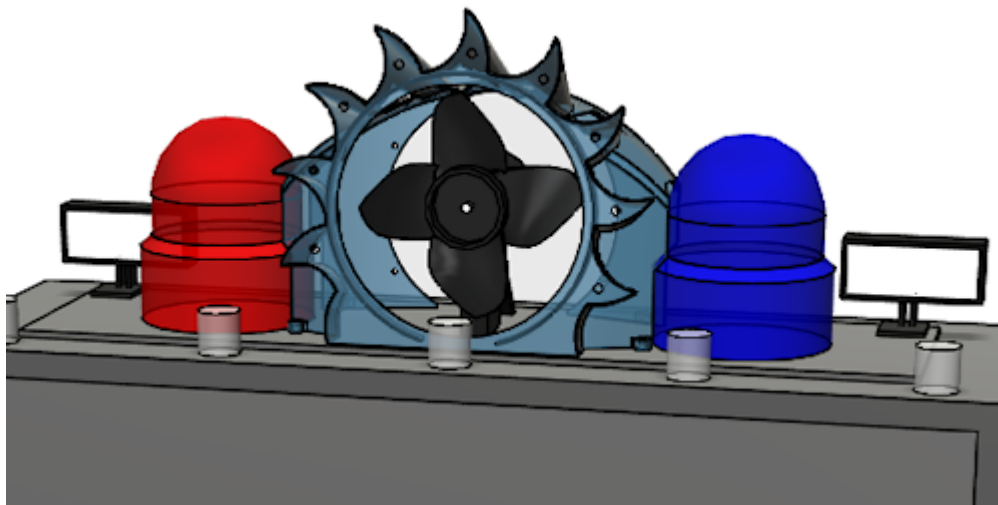
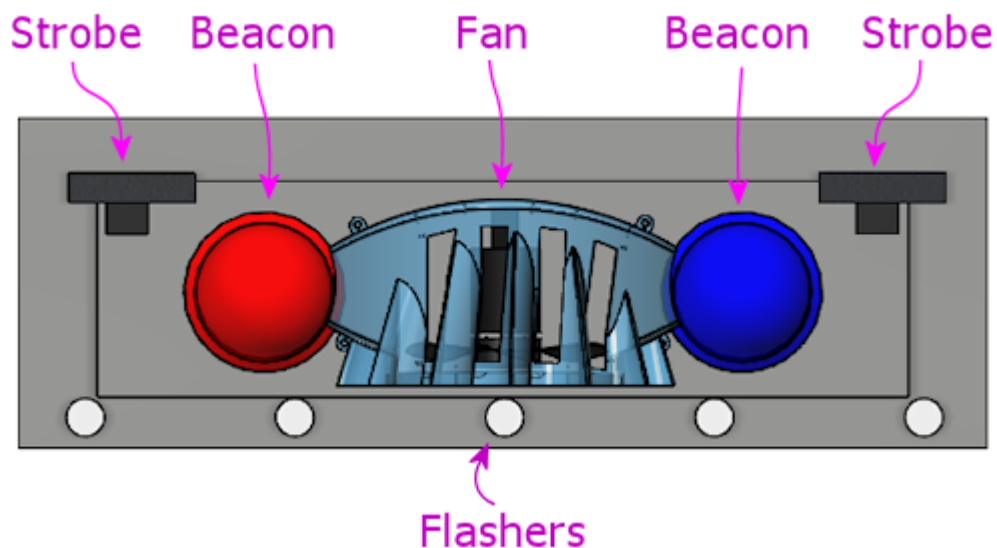


Some ideas for how to work your toppers into a theme:

- Brainstorm for natural real-world "topper" items fitting your theme, like *Taxi's* sign topper.
- Think about custom enclosures for one or more of the items. The fan is perhaps the most obvious enclosure to customize (like I did on my cab), since you need some sort of enclosure anyway, and it's easy to build around a bare motor-and-blade fan core. You could also create your own custom covers for beacons or strobes, or work them into a larger enclosure like a police-car light bar.
- Consider versions of the topper devices or exterior finishes that fit your theme. For example, if you're using a steam-punk theme, give your toppers a Victorian industrial look, with brass finishes and exposed rivets.

The Pinscape topper

The topper design that I came up with for my own pin cab is very specific to my theming, so I doubt it'll be of much interest to most pin cab builders other than as one more example. But in case anyone wants to reproduce it or use it as a starting point, here are some details about the design.



Most of the topper is made from off-the-shelf parts:

- A row of five standard pinball flasher domes with high-power RGB LEDs inside is arrayed across the front edge. These are simply wired in parallel with the main flasher board at the back of the playfield, so they always light up at the same time and in the same colors as the main flashers. See Chapter 56, *Flashers and Strobes* for details on setting up a standard flasher panel.
- A pair of Peterson 771 dome-shaped rotating beacons, one in red and one in blue. See Chapter 57, *Beacons*.
- A pair of "22 LED white strobes" that you can find on eBay. See Chapter 56, *Flashers and Strobes*.

The centerpiece is a custom-built fan with a 3D-printed plastic enclosure.

The fan is built with a generic 12V DC motor with a 1/4" shaft, mated to a press-on plastic blade. The blade is a part originally used in microwave ovens, Thorgren model number 6C2504C1, in black. This happens to be the same OEM part that Williams used for the topper fan in *Whirlwind*, so I was delighted to be able to find the exact same part. But don't worry if you can't find that particular blade; eBay has lots of similar fan blades that look just about the same. Just look for a 6" fan blade with a shaft bore that's the same size as your motor's shaft. A good search term to try is **6" fan blade**.

To mount the fan motor, I rigged a simple bracket using sheet metal. I don't have any tricks to suggest here; the bracket I came up with isn't particularly clever or elegant. You just need something that you can attach to the motor body to hold it at the right height above the backbox roof.

For more on building a fan like this from parts, and for details on how to wire a fan to your output controller, see Chapter 63, *Fans*.

The fan enclosure is designed to fit between the beacon domes, to create the impression that the enclosure and domes are connected. The area around the fan opening is meant to suggest a blazing sun, in a sort of cartoonish style.



You can download 3D plans for my fan enclosure in STL format here:

mjrnet.org/pinscape/downloads/PinscapeFanEnclosure.zip

Note that the ZIP file contains "front half" and "back half" models in addition to the full model. I created the half models to fit the limits of the specific manufacturing process I used, so these probably won't be useful unless you happen to use a printer with very similar constraints, but I included them just in case.

There's one more detail worth mentioning: the fan enclosure is decked out with lighting. (If you want to see it in action, I made a short video: www.youtube.com/watch?v=wJ1czPnjvDQ.)

The fan enclosure has two types of lighting. The first is one of those common 5050 RGB LED strips, installed around the perimeter of the front fan opening. This is the same type of LED strip normally used for Chapter 58, *under-cab lighting*. If you look

carefully at the fan enclosure model, you'll see a little lip on the inside of the opening, about 1 centimeter deep. That's the mounting surface for the LED strip. The strip is mounted on the inside of this lip, so that the LEDs face inwards towards the center of the opening. This creates a nice ambient light effect inside the fan. I wired the LED strip straight to my undercab lighting, so it shows the same colors as the undercab lights. That creates a nice effect with the output in DOF set to show the "undercab complex" effects, which changes the LED strip colors in response to game events. It makes the fan interior lighting fairly dynamic.

The second bit of lighting installed in the fan consists of nine high-power RGB LEDs, arranged around the perimeter of the opening, facing forward. Again, looking carefully at the STL model, you'll see a small circular hole in each of the "points" of the star-burst shape. These holes are the right size for typical 3W RGB LEDs, of the same type commonly used for RGB flashers, but in this case, *without* the aluminum "star" base that's usually used for the flashers. You can find these LEDs on eBay by searching for "3W RGB LED" - they look like this:

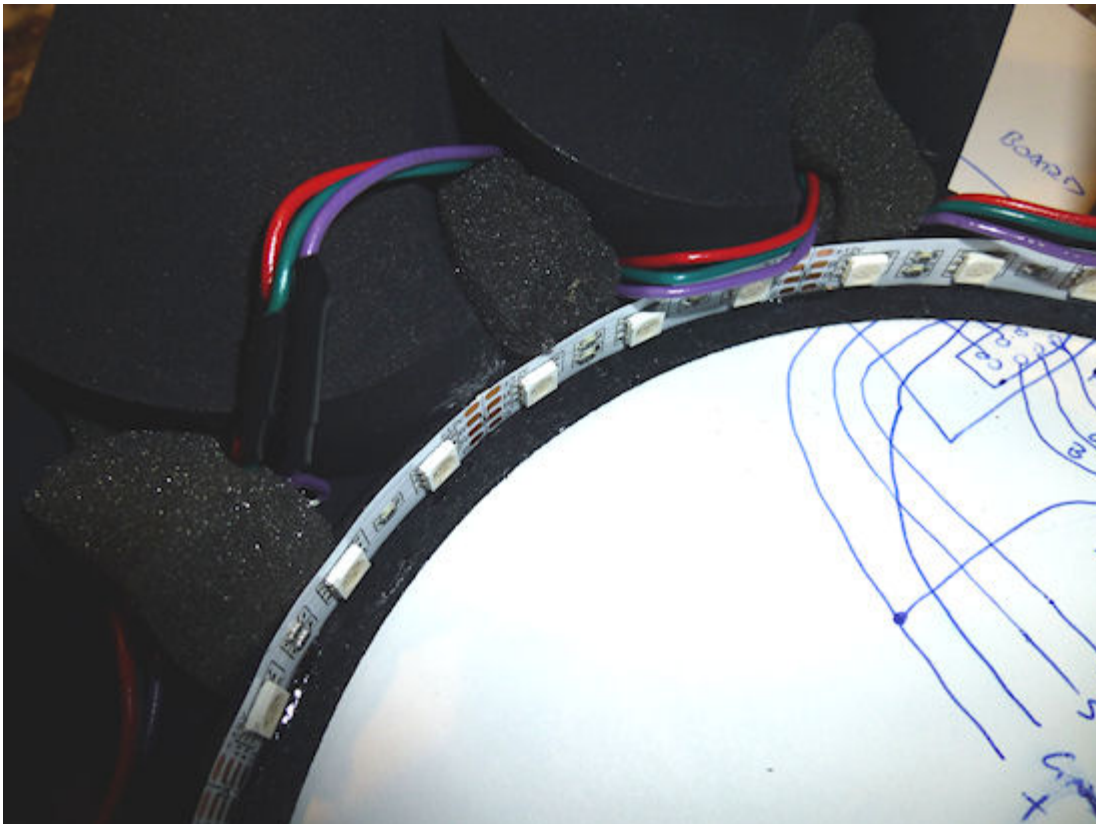
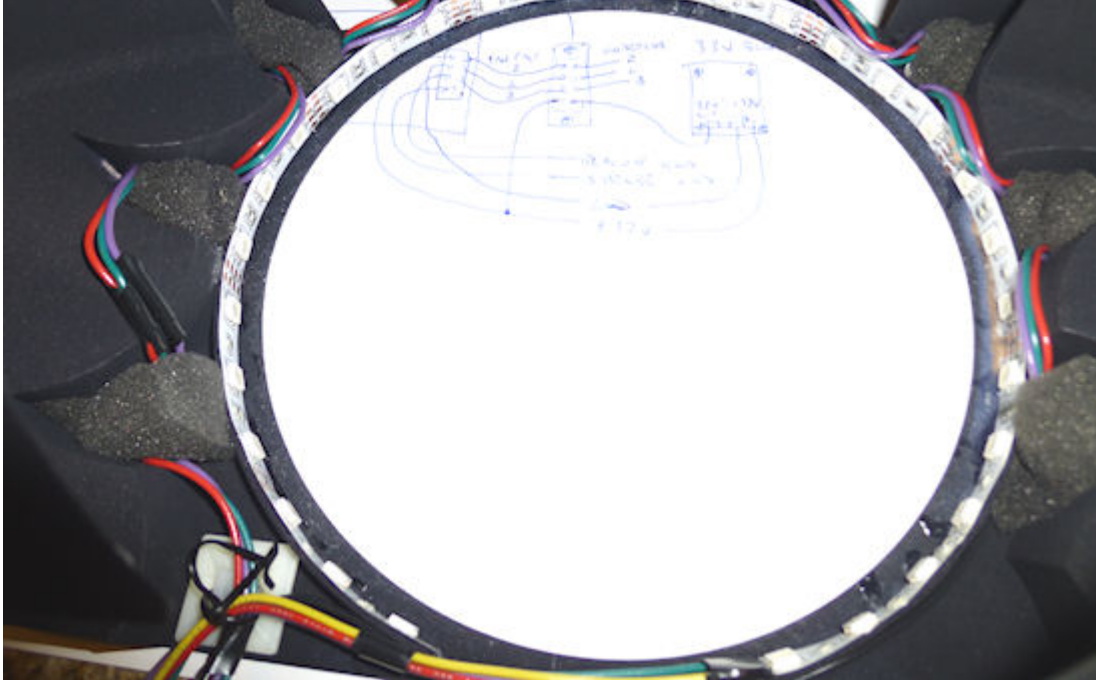


These are actually the same LEDs used in the "star base" type. The only difference is that they're sold as bare LEDs, not mounted to the aluminum heat sink base. The procedure for wiring them is the same as for wiring the flasher LEDs; see Chapter 56, Flashers and Strokes for more on that.

The method I used to mount the LEDs in the fan enclosure is inelegant and a little tricky (it takes some manual dexterity and a bit of patience), but it worked well and has held up well over the several years since I built it. I started by soldering hookup wire to the LEDs, with just enough wire between adjacent LEDs to reach from one hole to the next. The LEDs are wired **in series**, meaning that the "+" side of one LED connects to the "-" side of the next LED, and so on down the chain. There are no resistors in this chain - just the wires and the LEDs. Once all of the wiring was soldered, I arranged all of the LEDs into the desired locations, poking out through the star-burst holes. This is the part that requires dexterity and patience.

And now for the part that's truly inelegant. To secure the LEDs in place, I stuffed some packing foam into the pockets behind the LEDs. I tried some other approaches, the best hope being 3M VHB tape (which is pretty amazing for many similar applications), but that didn't work; it's hard to get anything adhesive to stick to 3D-printed plastic, since the surface tends to be uneven and powdery. The packing foam turned out to work surprisingly well, and it has the nice feature that it's easy to remove if any of the LEDs ever needs to be replaced or if a solder joint ever breaks.





Controlling the 3W perimeter LEDs is almost a whole separate project. There are two main parts:

- The first is the power supply for the 3W LEDs. Wiring LEDs in series means that you need to supply them with a voltage that's higher than the **sum** of the V_F ("forward voltage") values for all of the LEDs. For 9 of the 3W RGB LEDs, this works out to about 33V. So I used a DC-to-DC step-up buck converter to convert power from a 12V supply to 33V. (You can find such step-up converters on eBay; they run about \$10 for the size needed here.)
- The second part is something to control the LEDs. You could just wire them to your DOF output controller, and assign them to one of the existing DOF device types, such as the strobes or under-cab lights. I wanted something a little unusual, though, which required some more custom electronics. What I wanted

was to coordinate the LEDs with both the strobes and the beacons: when the strobes fire, I want the LEDs to flash white, and when the beacons run, I want the LEDs to flash rapid red and blue patterns like a modern police cruiser. You might be able to produce something like this with DOF directly, but it seemed easier in this case to build a little microcontroller project instead. As usual with microcontrollers, GPIO pins provide the connections to the outside world: GPIO input pins connect to the DOF outputs for the strobes and beacons, so that the controller can monitor DOF activations on those devices, and GPIO output pins connect to the fan LEDs, via MOSFETs. I wrote a small custom program for the microcontroller that watches the input connections from DOF, and when it sees one of them activate, it generates the appropriate light show on the fan LEDs.

The ZIP file linked above (with the 3D design for the fan enclosure) contains a hand-drawn schematic for my controller circuitry, and the C++ control program for the Trinket. The program is designed to be compiled and downloaded into the Trinket with the Arduino IDE. I apologize for the rough appearance of the schematic; this is directly from my original working notes, and I haven't had a chance to clean it up into a proper presentation.

Reviewing this circuit plan with fresh eyes, I see a couple of changes I'd suggest, if you plan to deploy this in your own cab:

- Add a diode (1N4007 should work nicely) in series between each optocoupler cathode (pin 2 of the PC817) and the DOF output controller port, with the striped end of the diode on the DOF port side. This will avoid any danger of feeding back the beacon/strobe supply voltage into the optocoupler LED. (LEDs don't typically have very high reverse voltage tolerance on their own.)
- You *might* need to add a capacitor, connected across the Trinket power supply pins, to filter electrical noise from the rest of the system. You'll know this is necessary if the Trinket randomly resets or behaves erratically; if it's stable, don't worry about this. If you see any glitchy behavior, try adding a 0.1uF capacitor with its leads connected to the Trinket's BAT and GND terminals. Position it as close to the Trinket as practical. If erratic behavior persists, try different size capacitors, even up to large sizes like 1000uF. Sometimes two capacitors in parallel work even better than one, such as a 0.1uF and a 100uF. (Larger capacitors, 100uF and above, are usually electrolytic, which have a "+" and "-" side, so be sure to connect the marked "-" lead to the GND terminal. Smaller "disk" capacitors aren't polarized. See Chapter 73, Capacitors for more if you're not sure.)

Note that my circuit design doesn't use conventional current-limiting resistors for the LEDs. Instead, it uses a feedback loop on the MOSFETs to throttle the current through the LEDs. I did it this way mostly because I was curious about how to create a current-limiter circuit like that. I don't think it's all that much better than the simpler approach with ordinary resistors, since ultimately it's just using the MOSFETs as variable resistors and burning off the extra power as heat, just as fixed resistors would. But it seems to work nicely, and it does have the slight advantage that you don't have to figure out the right resistor size for each channel; the feedback circuit amounts to a little analog computer that does the math for you each time you apply power.

43. Finishing Touches

Should you ever find yourself in a position where you're finished with your virtual cab and can't think of any more work to do on it, here are some miscellaneous ideas for extras, Easter eggs, and final polishing work.

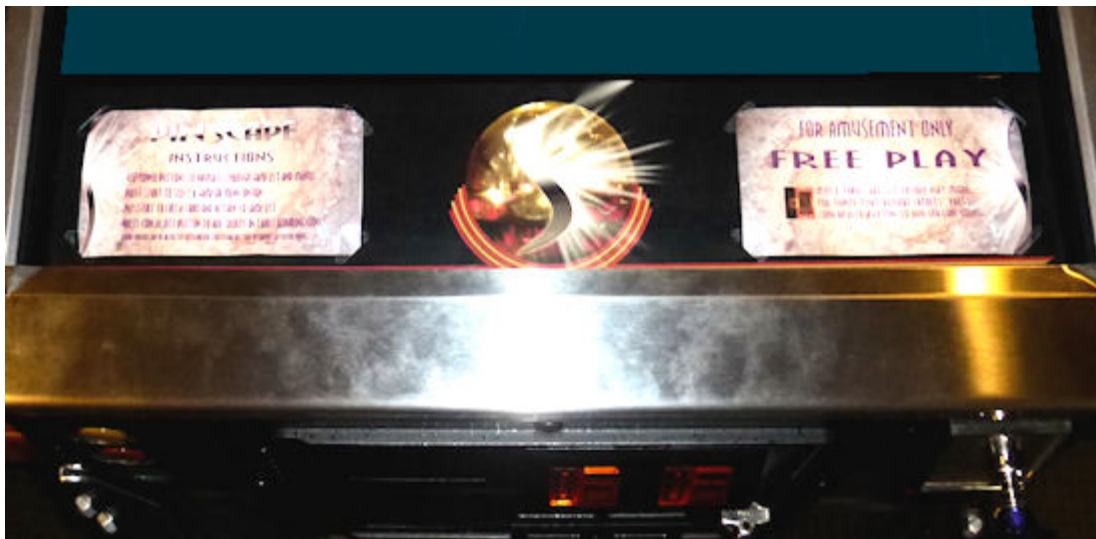
Apron

If you installed your playfield TV with a setback from the front of the cabinet to make room for the plunger, you'll need something cosmetically appealing to fill the space. My preference is to make this area resemble a pinball "apron", since that's what you'd find in the same space on a real machine.



The apron from Space Station (Williams, 1987). This is a fairly generic apron, with the Williams logo and some simple stripes for decoration. Most aprons from the era used the same overall shape and size, but many from the 1990s had custom graphics tied to the game theme in place of the generic logo and stripes.

We can't *precisely* replicate this look for a virtual cab, since we don't want that "V" shape that funnels the balls into the trough at the center. We just need a simple rectangular shape, to fill the area between the front of the cab and the bottom of the TV. But we can still duplicate the whole bottom section, which has the most distinctive visual elements anyway - the card inserts and center logo graphics.



The "apron" from my virtual pin cab, filling the space between the front of the cabinet and the TV.

Live card monitors

If you want to get really elaborate with the apron, replace one or both of the cards with displays. Real pinball instruction cards are around 6" wide by 3.25" high (that's the exact size for Williams games from the 1990s, and other manufacturers used slightly different sizes in the same range), which is very close to the size of a 6" diagonal 16:9 display. And you can actually buy LCD screens in that size - they make them for microcontrollers like the Raspberry Pi.

This is an extremely uncommon feature (I've only heard of one cab builder who's implemented it), so it's not all that well supported in the virtual pinball software suite. My own PinballY front end has built-in support for it, but that's about it as far as I know. I think it would be pretty easy to come up with your own custom software to keep it in sync, though, since all you need to do is display a static image each time a game is started.

How to construct an apron

On the real machines, the aprons are made of sheet metal, and typically have a smooth satin finish with silk-screened graphics or decals.

The virtual cab equivalent is such a simple shape that you can easily make it from a thin piece of plywood or particle board. However, it can be hard to get the right finish that way, since you'll probably end up with visible wood grain (even if you cover it with a decal for the graphics).

Another option is to make it out of sheet metal, if you have the tools for that. The finish is easier to get right, since the real ones use sheet metal as well.

I made mine with laser-cut acrylic. That let me cut the diagonal openings to hold the corners of the cards, to better match the look of the real machines. I faced it with a custom-printed decal - the same sort used for the cabinet itself. The combination of acrylic and the decal has exactly the right finish to match the originals.

Here are the dimensions I used for the corner cutouts. These are sized for the 6" x 3.25" instruction cards that Williams used on their machines in the 1990s. I'm leaving out the overall dimensions since you'll almost certainly need to adjust it to fit your cab. The overall width should be just slightly (about 1/4") less than the inside wall-to-wall width of your cab, and the height should be about the same as the distance from the lockbar receiver to the TV. (You want the lockbar to cover the bottom of the apron, for a seamless appearance, with no gap between the lockbar and apron.) Position the two instruction cards as you see fit, but I'd balance them so that they're at the same inset from their respective outside edges.



Instruction cards

Instruction cards on the real machines vary by manufacturer, so you don't have to match an exact size to look authentic. For reference, the Williams cards from the 1980s and 90s are 6" wide by 3.25" tall.

Traditionally, the left card has playing instructions, with a summary of the game's rules, and the right card is the pricing card, with the price per game, number of balls per play, and the obligatory "For Amusement Only" disclaimer.

You can find many examples of instruction and pricing cards for the real machines online by searching for "pinball instruction cards". There are some sites with collections of the original cards and fan-made custom versions. The original manufacturer cards were usually printed in black text on a white card, usually in a pretty plain style with minimal (if any) graphics. A lot of collectors these days like to replace the boring originals with fancier custom versions featuring full-color graphics based on the game's theme, so a lot of those are in circulation as well.

For a real game, the instructions card summarizes the scoring rules for the game; for a virtual cab, you'll probably want to summarize how to operate your game selection menu system instead. And the pricing card is usually a simple "Free Play" announcement. To give you some more specific ideas, here are the cards I created for my cab. You won't want to use them as-is, since they're so customized for my graphics theme, but maybe they'll be helpful starting points.



Sample virtual cab instruction card. The instruction card is traditionally displayed on the left side of the apron.



Sample virtual cab pricing card. The pricing card is traditionally displayed on the

Maintenance labels

It's a really good idea to label anything on the inside of your cab that's not pretty obvious just looking at it. That'll be a big help later on if you ever have to repair anything or if you want to make any upgrades. Do this while you're building the machine or shortly after you're done, while things are still fresh in your mind. Some things that are helpful to label:

- Wire harnesses and pluggable connectors. Put a label on each connector saying what it's for and where it plugs in.
- Wires going to screw terminals. Mark which terminal they connect to.
- Power supplies. Label each one's main function ("PC power supply", "feedback device 24V supply", etc).
- Power supplies with adjustable voltages, and adjustable step up/down voltage converters. Label with the purpose ("shaker motor speed control") and the voltage that you've determined as the correct setting.
- High voltage warnings. Put a high voltage warning label near any exposed wires or terminals with anything above about 24V. A general "High voltage inside" warning might also be appropriate; the real machines feature a label like that on the lockbar receiver.

⚠ CAUTION HIGH VOLTAGE Under Playfield TV

- Other warnings. Label anything else that you know to be hazardous, so that if anyone else tries to do any work inside, they'll know where they need to be careful.

Install 0 balls

This is just a little pinball-nerd joke. On the real machines, there's always a label on the top of the lockbar receiver (the metal part on the inside of the front wall that latches the lockbar in place) telling the operator how many pinballs the game takes, in case they were removed for transport.



"Install 4 balls" label, from Theatre of Magic (Bally, 1995)

For a virtual pin cab, the right number is, of course, zero. It makes a cute Easter egg for your pinball collector friends when you're giving a behind-the-scenes tour.

INSTALL 0 BALLS

16-8978-0

Here's the Math Nerd variation. It's based on the assumption that the largest "imaginary" number of balls you'd ever have to install would be the number needed to accommodate the virtual version of *Apollo 13* (Sega, 1995), with its rather silly 13-ball multiball mode.

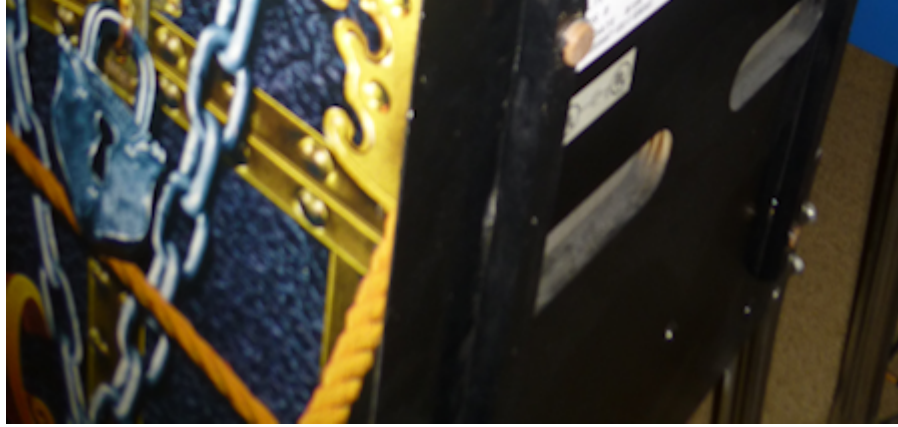
INSTALL 13*i* BALLS

16-8978-131

Manufacturing label

Real machines have a product label on the outside back of the main cabinet, at top left, listing the manufacturer, model number, serial number, etc.





The manufacturing label on the back of a 1990s WPC machine, at arrow. (The other two stickers are a UL approval label and an FCC approval label for radio-frequency emissions. I wouldn't advise imitating official certifications; let's just say that we'd like to avoid any Imperial entanglements.)

Here are a couple of mock-ups of the layout of these labels, to use as a model for creating your own. The styles are typical of the WPC machines of the late and early 1990s, respectively. These are usually printed at around 4"x4" to 5"x5".

PINSCAPE

MANUFACTURING COMPANY

SEATTLE, WASHINGTON, USA

AMUSEMENT MACHINE

SUITABLE FOR INDOOR USE ONLY

MODEL 10010 Pinscape

S/N



0000001

VOLTS 120 60Hz

MFG DATE 08/19/2019

WARRANTY VOID IF REMOVED



PINSCAPE MFG. LTD

SEATTLE WASHINGTON USA

AMUSEMENT MACHINE
SUITABLE FOR INDOOR USE ONLY

MODEL	PINSCAPE 10010
S/N	10010-00001
VOLTS	115VAC 60HZ GROUNDED OUTLET
AMPS	8.0
MFG DATE	08/19/2019

WARRANTY VOID IF REMOVED

16-8587-957

If you want to be really thorough, there's usually a second copy of the manufacturing label on the inside of the cabinet, typically stuck to the right wall near the front. In addition, a small warranty sticker (also with the serial number) is often affixed to the front of the cabinet below the coin door.

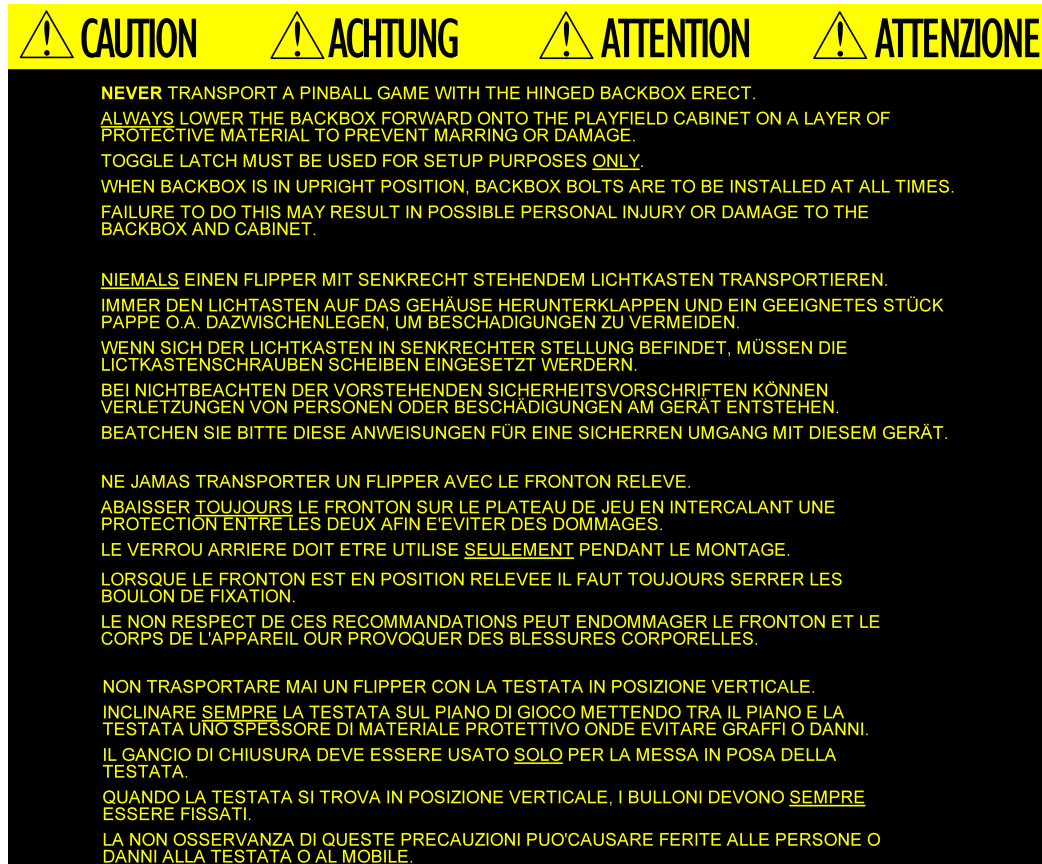


Original manufacturer's warranty sticker with the machine's serial number, typically placed on the front of the cabinet under the coin door.

WARRANTY VOID IF REMOVED	
S/N	10010-00001
PINSCAPE	PINSCAPE MFG. LTD

Backbox warning label

Most pinball machines have a warning on the back of the backbox cautioning operators to fold down the backbox for transport. The typical 1990s label is shown below. You can reproduce it (or something similar) for another bit of added authenticity.



This is normally printed at 18" wide by 15" high, on the back of the backbox (the exterior side), roughly centered. On the original machines, it was screen-printed, but that's prohibitively expensive for most pin cab builders. A more economical alternative is to use a printed vinyl decal as with the side art. Pre-printed decals with the standard warning text can be found online; search for **pinball backbox warning decal**. If you're having custom decals made for your side art, you can probably have a custom copy of the backbox label made at the same time.

Custom plunger knobs

This mod is popular with pinball collectors these days. Several small companies make themed knobs for specific titles, such as the Thing hand knob for *The Addams Family*, or a golf ball knob for *No Good Gofers*. If you based your cab's artwork on a particular real pinball title, a custom knob designed for that machine would fit nicely. There are also lots of ball-shaped knobs with metallic finishes, powder-coat finishes, and hand-painted patterns. You should be able to find something that matches just about any art theme.

For any of those options, search the Web for "custom pinball shooter knob".

In keeping with the DIY spirit of the pin cab project in general, I like the idea of

creating your own completely original shooter knob. This is actually pretty easy thanks to the availability of "knobless" rods. These are standard plunger rods, just without knobs at the end. Drill a hole in whatever you want to use as the knob, and epoxy it to the end of the rod. Combine this with 3D printing and you can fairly easily create a custom shooter with any theme-appropriate toy for the knob.

The knobless rods are available from any of the big pinball supply vendors; look for part 20-9253 (with no additional number like "-1" at the end) or search for "knobless shooter rod".

Lighted knobs

You can also find knobs with embedded LEDs that light up the knob from within. The ones I've seen are based on the modern Williams-style ball shooter assembly, so they should work on a virtual cab just as well as on a real one. Search for "illuminated pinball shooter rod".

Music jukebox mode

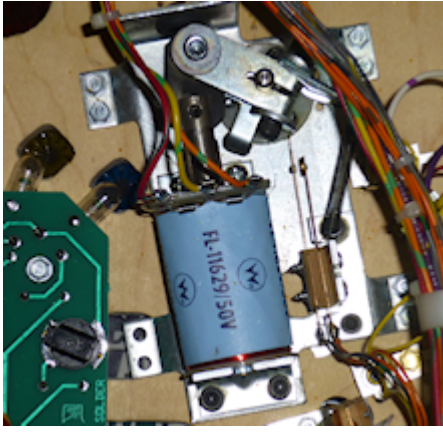
If you installed a decent set of speakers and amps in your cab, you can get double duty out of them by turning your cab into a music jukebox when it's not playing pinball.

I don't have any particular recommendations for jukebox software, but any Windows music player with a visualization mode should work.

Most people who do this set up the music player as a "system" under their menu system front end - that is, as though it were another pinball program like Visual Pinball or Future Pinball. A "system" in the front ends serves as an association with a Windows program (an application .exe file) that lets you launch that program to run any game listed with the system. Then you just create a pseudo "game" that represents the player and add it to your game list. To launch into music mode, you just navigate to the pseudo-game entry and launch it like any other game; that fires up the music program that you associated with the system.

For specific recommendations for music players and details on setting them up with the front ends, search for "jukebox" on the vpforums pin cab forum.

Part Three. *Feedback Devices*



44. Feedback Devices Overview

Feedback is probably the most challenging thing to set up in a cab. It requires coordination among several separate pieces of software and a specialized output controller device, all of which can be balky and hard to debug, and it also requires a certain amount of improvisation to create the various tactile and lighting effects. But this is one of those difficult jobs that's worth the effort. Pinball is an inherently physical, mechanical game that engages the sense of touch as much as sight and sound, so it really improves the simulation to bring some tactile elements into it.

Even though it's not exactly trivial to set up a good feedback effect system, it's not nearly as hard as it used to be. There was a time when these systems were elaborate, ad hoc contraptions, every one unique. That's no longer the case. Those early ideas have been refined and improved to the point where there are solid, repeatable recipes that you can follow. There's still plenty of room for creativity and experimentation, and you'll still have to do a little work figuring out what parts to buy, but I can give you detailed pointers on what you need and where to find it.

The next several chapters aim to make the big job of setting up a feedback system approachable, even if you're new to all of this. We'll try to cover all of the details: how to set up the PC software, what to buy for an output controller, how to install it, how to wire the devices to it, and how to build devices for all of the popular effects.

How feedback works

The whole idea of feedback devices is completely outside of our everyday experience as PC users, so if you're new to pin cabs, you probably have a lot of questions about how this works. I'll try to answer the big ones here.

How can a PC control all this stuff? Lights, motors, solenoids, etc? Using a special hardware device called an "output controller". You have to buy one of these (or build one, if you're using Pinscape) and connect it to your PC, usually with a USB cable. The controller is the bridge between the PC and the physical devices. The controller has wiring terminals, called "ports", where you connect the lights, motors, and solenoids. Once the feedback devices are wired to the ports, and the output controller is connected to the PC, software on the PC can send commands to the output controller to turn the attached feedback devices on and off or to different intensity (brightness) levels.

You can find a list of the available output controller options, with a detailed comparison of features, in Chapter 12, I/O Controllers.

Does that output controller need device drivers? Not in most cases, but it depends on which type of controller you're using. The LedWiz, PacLed, and Pinscape don't require any device drivers; the SainSmart relay boards do. If your device needs drivers, its documentation should cover that.

How many different feedback devices can I have? There's no upper limit as far as the PC is concerned, but each controller has its own limit for how many devices it can connect to. A single LedWiz can control 32 devices. Most of the SainSmart relay boards can handle 8 outputs. The Pinscape expansion boards can handle 65 outputs in the typical setup, but you can add modules for up to 128 outputs if you need more (which you probably won't!). You're not limited to a single controller, either, so if you need more ports, you can usually just add another controller. All of the controllers can coexist with one another, too, so you can mix and match them if you need different capabilities.

How do games know about my shaker motor? My flippers? Etc? That's handled in some software you run on the PC. There's a suite of software that almost everyone uses to make this all work, and it has a setup procedure where you enter information about your feedback devices. You tell the software if you have a shaker motor, flipper solenoids, flasher lights, etc, and you enter information on how they're wired to your output controller (more on this shortly).

Do I have to program every game with my devices? No. The device setup information is shared by all games, so you only have to enter this information once.

What if I install some devices but not others? That's fine. Games will use the devices you have, and won't care that others aren't present.

Do I have to program feedback into every game? No. Most VP 10 tables are pre-programmed to use feedback and will activate it automatically if your cab supports it. For VP 9, most games are pre-programmed to use feedback, but many require a couple of extra steps to enable it. We'll cover that in Chapter 46, DOF Setup.

Software and controller setup

There's a chicken-and-egg problem when it comes to deciding whether to install the feedback software or the feedback hardware first. To a certain extent, it doesn't really matter which one you work on first, because you really need both in place before either one will do anything. Even so, I recommend starting with the software, specifically DOF. That way, you'll be ready to do a practical test as soon as you get your first feedback device wired.

If you haven't already decided on an output controller, read Chapter 12, I/O Controllers. That chapter goes over the commercial and DIY options available for all of the I/O controllers. For our purposes here, focus on the output controllers, since that's what you need for feedback effects.

Once you've selected an output controller, proceed to the next chapter, Chapter 46, DOF Setup. That will help you set up the PC software that connects the pinball simulation to the hardware devices.

Next, set up your selected controller. If you're using a Pinscape controller, you'll want to read through The Pinscape Controller for instructions on building the hardware and setting up the software. If you've already done the basic setup work, you can skip straight to Chapter 111, Pinscape Feedback Outputs.

If you're using one of the commercial options, it should come with instructions of its own. But those are often pretty sketchy, and never contain any information specifically about pin cab use, so we've provided our own setup guides for the most popular options:

- Chapter 50, LedWiz Setup
- Chapter 51, SainSmart Relay Board Setup
- Pinscape Outputs Setup

If you're using one of those, we'll walk you through the process of installing the device, setting it up in Windows, and wiring output devices to it. If you're using a different device, you'll need to rely on the manufacturer's instructions, but we do have some generic advice in Chapter 47, Feedback Device Wiring.

Types of feedback devices

Here's a quick overview of the most widely used feedback devices. You certainly don't have to limit yourself to these ideas; the software and hardware let you hook up anything you can think of. But the devices below have become a sort of standard set that the existing software will put to good use automatically, activating them in sync with relevant events during game play.

Chapter 55, Button Lamps. These make your various buttons light up under software control: flashing the Start button, turning on the Launch Ball button when a ball is in the chute.

Chapter 56, Flashers and Strobes. High-powered LEDs and lamps that create a light show much more impressive than a mere video display.

Chapter 57, Beacons. Rotating or flashing police car-type lights. These are another part of the light show, and they're also found on so many real machines that they add a bit of authenticity to many games.

Chapter 58, Undercab Lighting. These are usually mounted on the bottom of the cabinet to create a pool-of-light effect around the machine.

Chapter 65, Addressable Light Strips. These let you create more complex lighting effects by manipulating individual lamps in an array, essentially like pixels in a display.

Chapter 59, Flippers, Bumpers, and Slingshots. Simulate the sound and palpable thud of the solenoids in a real pinball machine: flippers, bumpers, slingshots, kickers, etc.

Chapter 60, Replay Knockers. Reproduce the unique "knock" effect that real pinball machines make on replays and other game events.

Chapter 61, Shaker motors. Create a dramatic tactile effect that makes the whole machine vibrate. Some real machines have the exact type of device, so for those, a shaker re-creates the authentic playing experience. It also makes a great added effect even for games that didn't originally have a shaker.

Chapter 62, Gear motors. These simulate the sound of the small motors found on many real machines to animate playfield elements.

Chapter 63, Fans. These re-create the backbox fans featured on a scant few games (*Whirlwind*, *Twister*), but like shaker motors, they make for a dramatic effect that enhances many games that never had fans in the real versions.

Chapter 64, Chimes and Bells. Re-create the mechanical scoring chimes used in nearly all machines from the 1960s and earlier. These can make re-creations of older tables feel much more authentic.

A ranking by importance

Here's my purely subjective, totally biased ranking of the relative importance of the devices. The "importance" is on scale from 1 to 10. Now, keep in mind that these aren't goodness ratings; they're just relative degrees of importance. "1" isn't meant as a negative review score and certainly doesn't mean a device is *bad* to have. "1" just means that I rank that device as among the least important. All of the devices are nice to have if you can afford the cost, space, output ports, and time to set them up.

If you want a fully decked-out cabinet, these rankings shouldn't matter to you. You should just install everything. But if you're on a budget, or you want to start small

and add more as you go, these might help you prioritize. Again, though, these are just my opinions, and are not by any means the official consensus of the pin cab community.

Device	Importance
Shaker Motor	10
Flipper Solenoids	10
Flashers	9
Fan	9
Bumpers	8
Slingshots	8
Chimes/Bells	8
Replay Knocker	7
Strobes	5
Addressable Light Strips	5
Button Lamps	4
Beacons	4
Undercab light strips	3
Gear Motor	1

A few of these deserve an explanation.

I rank the shaker motor and fan so highly for the same reason: they both add a dramatic, tactile effect that goes way beyond "video game". Even after playing a lot of games on my cab, I still find these effects particularly engaging because they extend the game's reach beyond sight and sound.

Flasher lights are also at the very top of my list. They add visual impact that video can't approach. Real pinball machines have always used lighting to attract players and add to the playing experience, and this became even more important in the solid state era, where the software running the game could create complex lighting effects coordinated with the game action. Simulations reproduce the original lights in video form, of course, but video just isn't bright enough to create the same dramatic effects as real flashers. Flashers go a long way towards making it feel real.

The various solenoid effects - flippers, slingshots, bumpers - all rank near the top because these mechanisms are so central to real pinball. If you think recorded audio does these justice, you probably haven't played a real pinball machine in a while. The coils on the real machines are seriously strong. You don't just hear them, you feel them; they give the whole machine a jolt every time they fire. If you want to re-create the real playing experience, you really need to simulate that palpable jolt. The flippers, bumpers, and slings are all important, but the flippers are easily the top choice if you have to pick only one type. They're the ones you actually interact with constantly in every game, so the tactile feedback they provide is particularly

noticeable and particularly enriching to the experience.

The replay knocker and chimes are close on the heels of the other solenoid devices for all the same reasons. Recorded audio simply can't do these percussion instruments justice. I personally find the knocker to be more important than its frequency of use would suggest. Its importance comes from the fact that whenever it goes off, something great just happened in the game; the added sensory effect of that inimitable hammer strike really adds something. And if you're a fan of older electromechanical games, the realism added by chimes will make a world of difference.

The button lamps come in relatively low on the list mostly because you could get almost the same effect by just wiring the buttons to be constantly on. But it's still nicer to have them under software control, especially the Launch Ball button, which shows useful information on the game state in some games.

The gear motor ranks so low because it's just there for the sound effect, and this sound (unlike knockers and chimes) actually can be reproduced fairly well on the audio system.

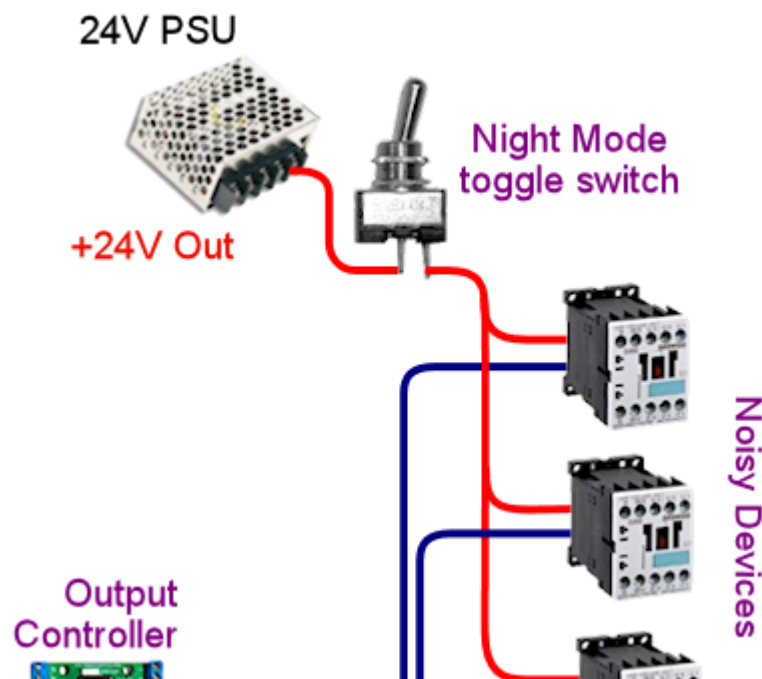
Night Mode

Many cab builders like to include a "night mode" switch, to disable the noisy tactile devices so that they can play during evening hours without disturbing housemates or neighbors.

If you're using a Pinscape controller, this is built into the software. See Chapter 113, Pinscape Night Mode for how to set it up.

Some of the plug-and-play feedback kits from Zeb's Boards also include this feature.

None of the commercial controllers have a native Night Mode feature, but you can set up your own DIY night mode switch pretty easily. The basic idea is to add a switch into the power supply circuits that feed power to the noisy devices. Set up the switch to control the "+" voltage going to the selected devices. When you turn the switch off, it cuts power to the noisy devices. For devices that don't need to be disabled at night, simply bypass the switch and wire them directly to the power supply.





If you need to include devices at different voltage levels in the night mode switching system, you'll need a "multi-pole" switch. A multi-pole switch is essentially a bunch of separate switches built into a single housing and controlled by a single lever, so that they all switch on and off together. You can use this to wire several voltages to the same switch, since the internal switches are all electrically independent even though they turn on and off together. If you need to control two separate voltage supplies, you'd need a double-pole switch. To control three voltages, you need a three-pole switch. You can find multi-pole switches from suppliers like mouser.com.

45. Power Supplies for Feedback

Feedback devices require power, so you'll have to provide power supplies in your cabinet for the devices you install.

All of the common devices run on low-voltage DC power. You'll probably have a mix of devices with different voltages, so you might need multiple power supplies to cover the different voltages. In this section, we'll help you figure out the set of voltages you'll need, and provide pointers for what to buy.

Common voltage levels

Here are the common voltage requirements for most of the feedback devices found in pin cabs.

Voltage	Devices	Source
5V	Flasher LEDs	ATX PSU or OEM 5V supply
6.3V	Round button lamps (#555 incandescent)	Step-down from 12V, or substitute ATX 5V
12V	Shaker motor Gear motor Fan motor Beacons Strobes LED strips Some solenoids	ATX PSU or OEM 12V supply
14V	Square button lamps (#161 incandescent)	Step-up from 12V, or substitute ATX 12V
24V	Contactors Some solenoids Chimes and bells	OEM 24V supply
30V-50V	Replay knocker Real flippers Real slingshots Real bumpers	High-voltage supply

Go through the list above and find the devices that you plan to install. For each one, note the type of power source required and add it to the list of components for your system.

You'll only need one power supply of each type, even if you have multiple devices that use it, because all of the devices that need a particular voltage can share the same power source.

Below, we'll go into the details on what to buy for each power source and how to set it up.

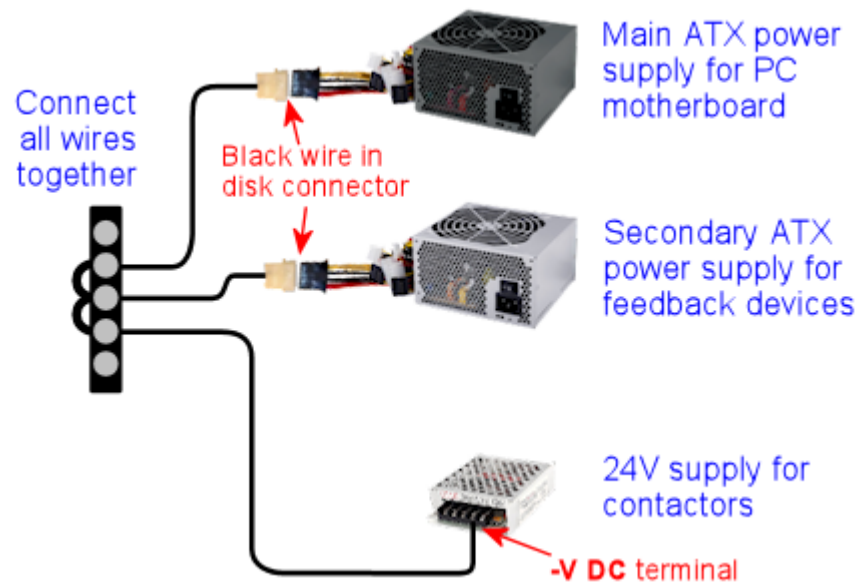
Interconnecting grounds

Whatever combination of power supplies you're using, there's an extra step you should take with each one: connect the "ground" terminals of all of your power supplies together.

"Ground" has several different technical meanings in electronics jargon, so let's make sure we're talking about the same thing. The ground terminals that I'm talking about interconnecting are the **negative DC terminals**. On ATX power supplies (the standard type of PC power supply), these are the **black wires** on all of the connector cables coming out of the box. On a power supplies with screw terminals, this is the negative DC output terminal, usually marked with one of the following:

- - (a minus/negative sign)
- **-V**
- **-24V** (or whatever voltage the supply produces)
- **0V**
- **G**
- **GND**
- **Ground**

In all cases, this is one of the power supply's **DC output** terminals, not anything on the AC input side that connects to the wall plug.



The point of interconnecting the power supply grounds is to ensure that all of the different circuits at all of the different voltage levels have a common reference point for 0V. This allows them to work together in a single system. This is important for the feedback devices because they connect to the output controller, which in turn connects to the PC (through the USB cable). Without a common 0V reference point, the common connections through the PC could allow hazardous current flows between power supplies. Interconnecting the grounds prevents this.

Note that it's not necessary to make any extra connections for step-up or step-down voltage converters. These are inherently connected to the common ground through their power input wires, as long as the power supply they're connected to is itself connected to the common ground point.

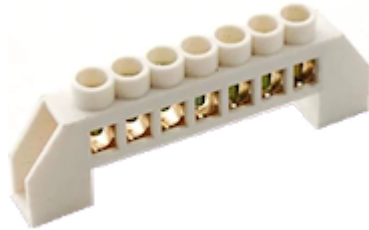
Wiring suggestions for ground interconnects

The important thing is to connect all of the grounds together electrically. Exactly how

you wire this is up to you.

The easiest way is to simply daisy-chain a hookup wire from one power supply ground to the next.

A somewhat neater approach is to install a "power distribution block" or "ground bar". This is basically a metal bar with a number of screw terminals for attaching wires; since it's one big metal bar, all of the terminals are electrically connected, so any wires you connect will all be shorted together, exactly as we want. Adafruit makes a nice power block that you can buy directly from their site or via retailers like Mouser.com: Adafruit 737. It looks like this:



You can find similar products at home supply stores and hardware stores, where they're usually called "ground bars". The Adafruit setup is a bit nicer because it has a plastic housing for easy mounting; the hardware store type is usually just a plain metal bar with screw terminals. In any case, both types provide screw terminals that are all connected together electrically. Simply connect a wire from each of your power supply's Ground terminals to a terminal on the bar.

Individual power supply descriptions

Now let's look at the common types of power supplies you can use in a pin cab.

ATX power supplies

ATX PSUs - the same type that you're probably using for your PC motherboard - are great for feedback power because they're cheap sources of the most common voltages you need, 5V and 12V. They crank out tons of power at a low price; the cheaper ones only cost about \$30 and provide 300W or more.



It might be tempting to take advantage of the fact that you *already* have an ATX PSU in your cab to power the PC motherboard, and give it double duty powering your feedback devices as well. But I'd recommend against that. You really should buy a second ATX PSU that you can dedicate entirely to your feedback system.

The reason to use a separate power supply for feedback is that many of the devices are electrically "noisy", meaning that they inject voltage spikes and dips into the power supply circuits when they operate. The mechanical devices in particular (solenoids, contactors, motors) can cause the power supply voltage to fluctuate quite a lot when they switch on and off, because of the sudden change in load. If you connect these devices to the PC's power supply directly, you'll subject your motherboard to that electrical noise and the fluctuating voltages. It's best to use a whole separate PSU to better isolate your motherboard from all of that and keep its power supply as stable as possible.

There's no need to match your ATX PSUs, by the way: you don't have to buy the same make or model for the secondary unit that you're using to power your motherboard. Any cheap, low-end ATX PSU rated around 300W of power should be perfectly adequate for feedback device power.

How big an ATX power supply do I need?

For powering feedback devices in a pin cab, most people are fine with a low-end ATX PSU rated at 300W or higher.

Figuring out the exact size of the ATX PSU you might need isn't easy, because the spec sheets for ATX supplies don't usually list the exact power limits at the different voltage levels (3.3V, 5V, and 12V). They only tell you the *total* power. These supplies always have lower internal limits on the individual voltages. Some manufacturers list the specs at that level of detail, but most don't; usually they just advertise the total Wattage. Fortunately, even a low-end ATX supply (300W, say) has enough power for the standard feedback devices, so in practice there's usually no need to figure out the exact specs you need. Just buy a cheap ATX PSU and it'll probably work fine. If you feel that you have an unusually power-hungry set of feedback devices, you might upgrade to something in the 500W range, but I think it's diminishing returns beyond that.

If you want to figure out exactly what you need, start by grouping your devices into 5V and 12V groups. Don't include devices that you'll power from separate power supplies, such as 24V contactors or 50V solenoids. (However, *do* include any devices that you'll connect indirectly to the ATX power supply via things like voltage step-up or step-down converter boards.) Figure out the "worst case" for how many of the devices in each group will be activated **simultaneously**. That tells you the maximum current (Amps) needed at any one time at each voltage. Once you know that highest simultaneous current draw, multiply it by the voltage to get the wattage. If you can find manufacturer specs for your ATX supply that tell you the individual power limits for the 5V and 12V rails, use that to make sure the power supply is big enough; if the specs don't include that level of detail, I'd make a wild guess that you can expect at least 30% of the rated total power to be available on each rail. So for a 300W power supply, I'd expect at least 100W to be available on the 12V rail, for a limit of at least 8A.

How to connect devices to an ATX power supply

The connectors on an ATX power supply are all designed to plug into matching connectors on PC motherboards, video cards, and disk drives. Pin cab feedback devices aren't equipped with the matching connectors, though, so we have to improvise a bit to break the ATX power supply out of its connector jail. There are three main ways to do this:

- Use a breakout board. This is the easiest way. A breakout board is a small circuit board that you can buy on eBay or Amazon that has a PC motherboard socket and a bunch of screw terminal connectors for the voltage outputs - ground, 3.3V, 5V, 12V. Plug the main 24-pin motherboard power cable from the power supply into the socket in the breakout board, and connect your devices to the appropriate screw terminals.

If you buy a breakout board, you can skip all of the details below about overriding the soft power circuit and wiring up your own disk-type connectors. Simply use the screw terminals on the breakout board to connect hookup wire between the devices and the breakout board.

To find suitable boards, search for **ATX 24-pin breakout board**. The easiest kind to use is the type with screw terminals for the voltage outputs. These are

currently about \$15 on Amazon. (Some boards use other types of plastic plugs for the outputs, which doesn't really help if you just want to use hookup wire directly.)

- Attach your own matching connectors to your feedback devices. This is a little more work than using a breakout board. Follow the steps below if you want to go this route.
- Snip off the disk connectors from the ATX power supply cables, and connect the devices directly to the exposed wiring, by soldering or using a screw terminal strip. This procedure is basically the same as creating your own matching connectors, except that you get to skip the connectors and just connect directly to the wires. Follow the steps below.

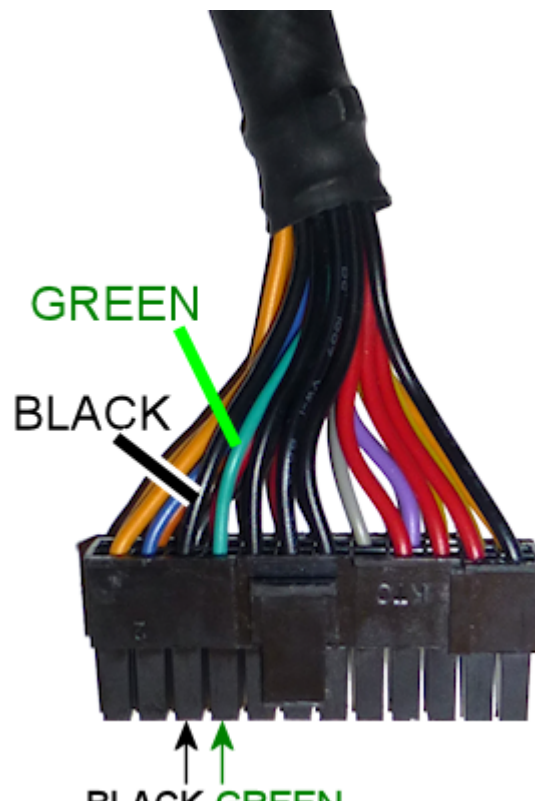
Overriding the soft power circuit

In order to use an ATX power supply with feedback devices, you have to override its "soft power" control circuit. This is a circuit inside the power supply that allows the computer operating system to switch the power on and off under software control. This is how Windows powers off your computer when you select "Shut Down" from the Start menu.

The snag this creates for our secondary ATX power supply is that the default condition is "power off". The motherboard has to send a signal to the power supply to turn the power on in the first place. With a secondary ATX supply, we're not connecting it to a motherboard at all, so we have to send this signal ourselves.

Fortunately, overriding the "power on" signal is extremely simple. It's just a matter of shorting together a particular pair of wires in the big 24-pin connector that you'd normally plug into the motherboard.

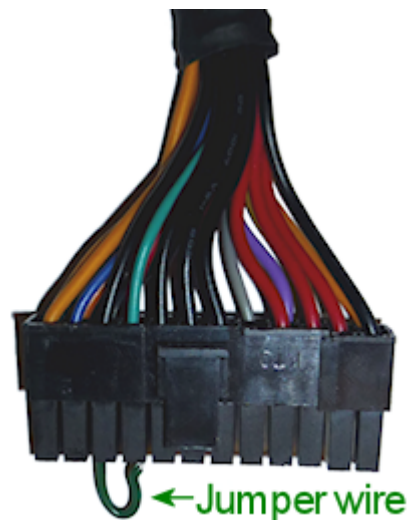
- Find the large 24-pin motherboard connector. On older units, this might be a 20-pin connector. See the illustration below.
- The wires to the connector are all color-coded. Find the **green** wire. There should be only one green wire, and it should be the fourth wire from the left if you're holding the connector as shown below.



- Find the **black** wire next to it to the left.
- Connect the green and black wires together.

Exactly how you connect these two wires is up to you. Here are some options:

- Use a piece of solid hookup wire around 22AWG in thickness, and about 1" long. Strip both ends. Insert the ends in the pin connector sockets for the black and green wires. Tape it securely in place with electrician's tape. This is simple and doesn't permanently modify the PSU, in case you ever want to return it to service as a regular PC power supply in the future. The downside is that it can be flaky. To improve reliability, use wire that's thick enough to fit snugly in the sockets without any play, and make sure it's inserted far enough that it won't work its way loose.



- If you don't mind permanently modifying the power supply, you can simply cut the black and green wires at the ends where they enter the connector plug housing, strip the ends, and solder them together. Wrap the exposed solder connection with electrician's tape to insulate it. This approach is simple, and it's more reliable than the jumper wire technique above, but it permanently modifies the cable. You won't be able to use the power supply as a regular PC power supply in the future.

You can test your wiring simply by plugging the power supply into AC power. If the fan turns on, your wiring worked, and the power supply will now be permanently powered on. If the fan doesn't turn on, check your wiring; if you used the non-permanent jumper wire technique, try jiggling the wire to see if you just have a loose contact. Also make sure the "hard" power switch (usually located next to the AC power cord) is switched on - that cuts the AC power input when switched off, so you'll want to leave that switch in the on position permanently.

If your jumper wire looks solid and the fan still won't turn on, or if it mysteriously shuts off after a few minutes, your power supply might require a minimum load to operate. More on this below.

Minimum load

Some ATX power supplies have a load sensor circuit that shuts off power if the computer isn't drawing at least a minimal amount of current. This is meant to prevent the power supply from operating when unplugged from the motherboard.

You probably won't have to worry about this, because most ATX power supplies don't have anything like this. The cheaper ones are less likely to have them than more expensive ones.

You can easily test for a load sensor by plugging your PSU into AC power, after overriding the "soft on" circuit as described above. If the fan doesn't turn on, you might have a load sensor that you'll have to deal with, but you should double-check the easier stuff first to make sure you're not on a wild goose chase: make sure the hard power switch on the back of the unit is switched on, make sure it's plugged in to a working AC outlet, and make sure your green-to-black jumper wire is installed properly (see Overriding the soft power circuit above).

If the fan is running, leave the PSU on for about five minutes without anything else attached. If the fan is still running, you probably don't have any sort of load sensor, so you don't have to worry about the rest of this part.

If the fan won't turn on at all or turns off after a few minutes, you probably have the load sensor. Like the soft-on circuit, you can work around this and force the PSU to operate, but the procedure is a little different. You can't just cross a pair of wires in this case; what you have to do is provide the minimum load that the sensor is looking for.

The easiest way to set up a minimum load is by installing a resistor between a **red** and **black** wire in the motherboard connector (the same connector that has the green wire for the soft-on circuit). Use a **10 Ω , 10W** resistor, such as a Xicon 280-CR10-10-RC.

As with the soft-on circuit, you can wire this to the ATX motherboard connector plug by inserting wires into the sockets, or you can clip the wires and solder them to the resistor leads. There are several black wires and several red wires going to the 24-pin connector; you can pick any of them, since they're all wired together inside the power supply.

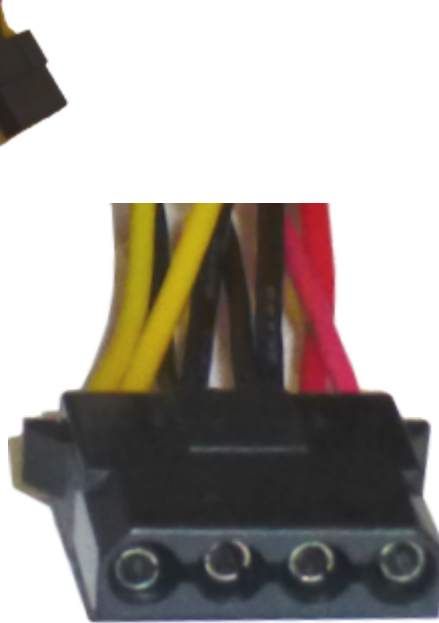
The 10 Ω resistor creates a constant load of about 500mA, or 2.5W. This should be enough to satisfy the load sensor on any power supply that has one. Unfortunately, the resistor simply wastes this power by turning it into heat, but 2.5W is a tiny fraction of the available power even for a very cheap, low-end ATX PSU. The cheapest ones supply about 300W, so wasting 2.5W won't make a noticeable dent in your power budget.

Note that the 10 Ω resistor will get pretty hot: remember that its whole purpose is to waste power by producing heat. You should mount it in an open area where it gets some airflow and where nothing else will come into contact with it, particularly wires (the heat could melt their insulation).

How to connect 5V and 12V devices

I recommend using the disk connectors to connect feedback devices. These are the large 4-pin female connectors, also often (incorrectly) called "Molex connectors", that look like this:





An ATX power supply typically has at least two cables attached with one or two connectors of this type per cable, in a daisy chain arrangement. You'll probably also find one or two of the thinner SATA power connectors on each cable as well.

I like using the large 4-pin connectors because they have a high current capacity (about 10A per pin), and they're fairly plentiful. They also use a standard plug format, so you can build a mating connector that you can simply plug in without modifying the PSU wiring.

The wires connected to these plugs are color-coded to tell you the voltage on each wire. The wire colors are standardized across all ATX power supplies, so they'll be the same no matter what brand you're using.

Wire Color	Voltage
Black	0V (Ground)
Red	+5V
Yellow	+12V

The easiest way to use these connections is to cut off the plug with wire cutters, strip the ends of the wires, and solder your own hookup wire to the ends, to extend them to the needed length to reach the devices you want to connect. (Be sure to cover the exposed wire and solder joints with electrician's tape for insulation when you're done.) You can then run your hookup wire to a terminal block for distribution to different devices, or you can run the wires directly to the devices that use them.

The slightly more difficult, but neater and cleaner, way to use these is to build mating connectors. The official brand name for the connectors is Amp Mate-N-Lok. Here are the parts you need to build a housing with crimp pins:

- TE/AMP 1-480426-0 4-pin housing
- TE/AMP 60620-1 male crimp pins, quantity 4 per housing

These are crimp-pin housing, so it's best to use a crimping tool to assemble them. See Chapter 82, Crimp Pins.

If you build the housing, you can attach hookup wire and run it to a terminal block for distribution or directly to the devices. So it gives you the same end result as cutting off the connectors and soldering the wires, but it's nicer because you don't have to modify the power supply wiring at all. You just plug in the connector.



I recommend using 20 AWG wire for these connectors, since this will fit the crimp pins and provide plenty of current carrying capacity (about 11A). You want a fairly high current limit for these wires, since they'll probably be carrying power to multiple feedback devices.

By the way, the two black wires going to this connector are both 0V/Ground connections, in keeping with the standard color coding. The reason there are two copies of the ground wire is that the extra wire doubles the current carrying capacity of the cable. The ground connection has to handle all of the current going through both the +5V and +12V wires to this connector, so it makes sense that they'd provide twice as much wire capacity for it.

Single-voltage OEM power supplies

You can find cheap, no-brand power supplies on eBay in a variety of common voltages needed in virtual pin cabs, including 5V, 12V, 24V, and 48V. eBay sellers often call these LED light strip power supplies, but they're really designed for sale to manufacturers who will incorporate them into finished products, so they're sometimes called OEM power supplies (for "original equipment manufacturer").



These units are your best option for voltages you can't get from an ATX PSU, particularly 24V for contactors and chime coils, and 48V for replay knockers and other pinball coils. You can also find OEM PSUs in 5V and 12V, but I prefer using an ATX power supply for those voltages. ATX supplies are usually cheaper for the amount of power you get, and they have a safer design.

As unbranded OEM parts, these units tend to be inexpensive, but by the same token, and they're not at all consumer-friendly. They don't come with any instructions, and they don't even come with AC power cords, since they're meant to be installed inside a product that provides one. You'll have to wire the AC line power yourself. That involves hazardous voltages, so if you're not somewhat comfortable with DIY electronics, you might want to find another option. You'll probably also have to improvise a protective cover for the AC power wiring, since these units usually have exposed screw terminals for the AC wires.

Where to buy

eBay is the place to buy these.

To find them, search eBay for the "24V power supply", or whatever voltage you're looking for. You should be able to find 5V, 12V, 24V, and 48V versions. They should look approximately like the picture above: bare metal cases with a set of screw terminals on the back. There are generally no switches, controls, or indicator lights;

there's sometimes an adjustment screw to fine-tune the output voltage, but that's usually it as far as controls go.

When you find items that match this description, do a little comparison shopping to find a good value. It's always important to comparison-shop on eBay, since some sellers set asking prices that are completely uncompetitive.

Choosing power capacity

In addition to the output voltage, you'll also have to choose the power capacity you need. Higher power is more expensive, naturally, so you're wasting money if you buy something much bigger than you need. Higher-power units also tend to be physically larger. However, you do have to be sure to get something adequate for your needs, or you'll overload the PSU. A properly designed power supply has protective circuitry that momentarily cuts power to the attached devices if it's overloaded, but I don't necessarily trust the cheap OEM supplies to have that protective circuitry built in.

To determine the power capacity you need, make a list of the devices you're planning to attach to the supply. Estimate how many of them will typically be activated *simultaneously*. Add up the current draw in Amps of the largest devices that will be activated at the same time.

For example, if you're using a 24V supply for a set of contactors, you could reasonably expect three or four of the contactors at most to fire at the same time (both flippers, a couple of bumpers). Each contactor draws about 500mA, or half an Amp, so four of them at once would draw 2A. To be conservative, I'd add 25% to 50% as a safety margin, so in this case I'd look for a PSU rated for 3A or higher.

eBay sellers will typically quote ratings in both Amps and Watts, but some will only give you one or the other. Fortunately, it's easy to convert in either direction. Use this formula:


$$\text{Watts} = \text{Volts} \times \text{Amps}$$

For example, if a seller tells you that a 24V power supply is rated for 120W, you can use the formula to calculate that it can supply 5A ($120\text{W} \div 24\text{V} = 5\text{A}$).

Wiring

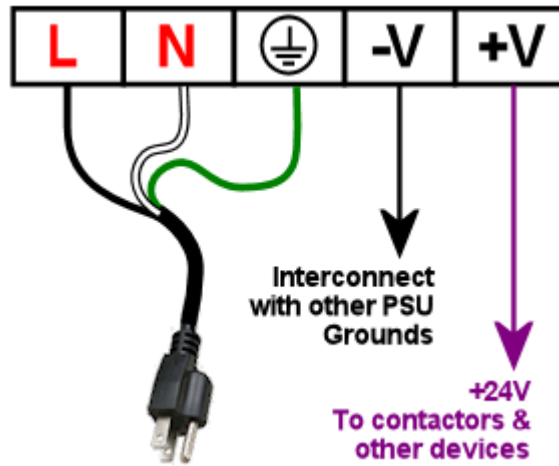
Wiring these power supplies is fairly easy. They usually come with screw terminals, so connecting hookup wire is just a matter of stripping a bit of insulation off the end, wrapping the bare wire around the screw, and tightening the screw.

The one snag is that they usually don't come with AC power cords, so you'll have to buy that separately. You can buy these at hardware stores and electronics stores, or you can use an extension cord or an old PC power cord if you have one lying around. To convert an extension cord or PC power cord, cut off the female end, cut off a few inches of the outer insulation (being careful not to cut the wires inside), and strip about 1/4" of insulation off the ends of the three inner wires.

The AC power cord's inner wires should consist of a black wire, a white wire, and a green wire. Connect the black wire to the power supply's "L" terminal, connect the white wire to "N", and connect the green wire to "G". The "G" terminal might instead be marked with the "ground" symbol ().

Here's a wiring diagram showing the typical markings on the generic eBay power supplies, and how to connect each terminal. Your unit might have different markings. If the markings are different and the correspondence with the diagram isn't obvious to you, check with the seller or ask for help on the forums. Connecting the AC power to the wrong terminals could be hazardous, so be sure you've identified the correct

terminals.



For safety, be sure to cover the terminals with an insulating cover after the wires are connected, and make sure that no bare wire is left uncovered. These power supplies usually come with a plastic cover for the screw terminal area, but if yours doesn't include one, you should improvise something to protect against accidental contact. The AC power cord wiring carries hazardous high voltage, so you want to make absolutely sure that you can't accidentally touch those wires while working in the cab, and also make sure that nothing else in the cab (including loose parts) will ever come into contact with them. Any short circuit involving the AC wiring could cause a fire or other severe damage.

24V power supplies

A few devices require 24V power supplies, particularly the Siemens contactors that many cab builders use to simulate flippers, slingshots, and bumpers. The coils in 1960s chime units also run on 24V, and you can also use 24V for the replay knocker, although I recommend using a higher voltage for that (see below).

The easiest way to get a 24V source is to buy a cheap no-brand 24V single-voltage supply on eBay. See OEM power supplies above.

If you're only using your 24V supply for contactors, a 3A/72W unit should be sufficient. If you're using it for a replay knocker and/or a chime unit, I'd look for at least 6A, and preferably 8A to 10A.

6.3V step-down converter

Most cab builders use the small round arcade buttons of the type pictured at right for the main front panel buttons: Start, Exit, Extra Ball. The standard type has an integrated lamp for illuminating the button. These are usually type #555 incandescent bulbs, which require an unusual power supply voltage of 6.3V.



If you're using these buttons, check the type of lamp inside. It might be an incandescent bulb or an LED. If it's an LED, it will run fine on 5V, so no special voltage is needed. If it's an incandescent #555 bulb, though, it's designed to run on 6.3V. (If you're not sure which is which, incandescents are the type with a visible wire filament inside a clear glass bulb or tube.)

If you have the incandescent type, there are three main options for how to deal with their special voltage needs:

- Ignore the special voltage and just use 5V from your ATX power supply. Many

cab builders do this because it's convenient. The bulbs will work with a 5V supply, but they'll be noticeably dimmer than with the 6.3V they're designed for. For many cab builders, the reduced brightness is an acceptable tradeoff for the convenience of using the existing 5V supply.

- Avoid the whole problem by replacing the bulbs with LEDs. Pinball and arcade supply vendors like Pinball Life sell drop-in LED replacements for #555 bulbs that fit the same sockets. Search for "#555 LED" at Pinball Life or your arcade supplier. The LED replacements should run equally well on 5V or 6.3V, with no significant change in brightness.
- Use a step-down voltage converter to convert 12V from your ATX power supply to 6.3V. Use the 6.3V converter output to power the bulbs. This allows the bulbs to operate at full brightness, but it's slightly more work (and expense) because it requires buying and installing the converter. The rest of this section explains how to set this up.

What to buy

You can find fixed-voltage 6V step-down converters at pololu.com.

If you prefer the variable voltage type, search on eBay for "DC to DC step down". This should turn up several small devices that look roughly like this:



These come in a variety of voltage and power ranges. They'll state voltage ranges like this: "7-32V to 1-28V". This means that the device accepts input voltages from 7V to 32V, and produces regulated output voltages from 1V to 28V. For our 6.3V bulbs, we need something where 12V is within the input range, and 6.3V is within the output range. So "7-32V to 1-28V" will work: our required 12V input voltage is within the quoted input range of 7-32V, and our required 6.3V output voltage is within the quoted output range of 1-28V.

Note that you should find one with a *range* of outputs ("1-28V"), rather than a set of discrete outputs ("3V, 3.3V, 5V..."). A range of outputs means that the device has an adjustment screw that lets you select any voltage in the range. A list of discrete outputs means that it has a switch that can only select among the listed voltages. We need the adjustment-screw type because we need to dial in an unusual voltage that won't be offered on any of the pre-selected switch types.

The device will also quote a power level, in Amps, Watts, or both. Each #555 bulb requires 0.25A, so if you have four such bulbs, you need $4 \times 0.25A = 1A$. This gives you the minimum; buy something rated for that much or higher. Most of the devices you find on eBay will be rated for much higher power levels than you need; you can buy anything that meets your minimum requirement.

How to wire the converter

These converters are simple to wire. They usually have four screw terminals with these markings:

- IN+
- IN-
- OUT+
- OUT-

The markings should be printed either on the top of the circuit board near the terminals, or on the bottom of side of the board directly under the terminals. Sellers sometimes include diagrams in the eBay listing page showing the terminal assignments, so check for that and make a screen shot of the page if you find it. That might come in handy later.

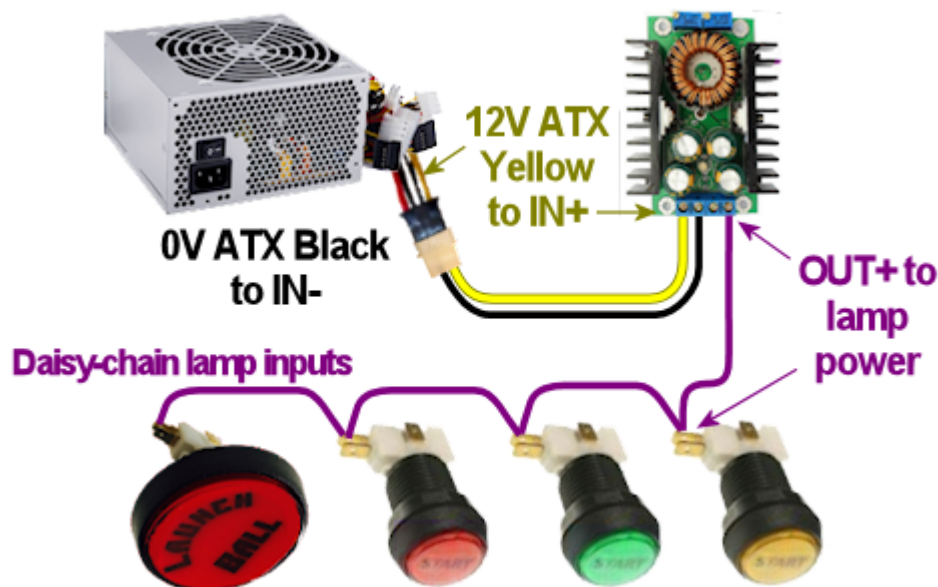
Once you identify the terminals, the connections are straightforward. Use hookup wire to make the following connections:

- IN+ connects to +12V (yellow wire) from your secondary ATX power supply
- IN- connects to 0V/Ground (black wire) from your ATX power supply (or, equivalently, you can connect it to the terminal block where all of your power supply grounds are interconnected: see Interconnecting grounds).

Before you proceed, you must adjust the output to the 6.3V we're after. You need a voltmeter for this step. Set your meter to read VOLTS. Connect the meter's leads to OUT+ (red lead) and OUT- (black lead) on the converter. Turn on the ATX power. Find the adjustment screw on the converter; this is normally a small slotted screw sticking up from a small plastic box on the top of the unit. Watching the voltmeter reading, turn the screw. Observe the effect on voltage. Adjust the screw until the output voltage reads 6.3V. Let it sit for a minute to make sure it remains stable. Once you have the right voltage dialed in, you can turn off power and put away the meter.

Now you can complete the wiring:

- OUT+ connects to one of the power terminals for each button lamp. Incandescent lamps aren't polarized, so both terminals are equivalent. You can daisy-chain this connection from one button to the next.
- OUT- can be left unconnected. (It's already be connected internally to IN- within the converter, so you don't have to wire it to anything yourself.)



If you're using large rectangular arcade buttons on your cabinet (for example, for front-panel buttons), these sometimes come with #161 incandescent bulbs to illuminate the buttons.

As with #555 bulbs, these bulbs require an unusual power supply voltage, in this case 14V. You can power these with the 12V supply that you probably have available from the ATX power supply, but they'll be quite dim if you do; they need 14V to operate at normal brightness.

You have three options with these, as with the #555 bulbs: you can accept the reduced brightness and use the conveniently available 12V; you can replace the incandescent #161 bulbs with LED substitutes, which should run at full brightness at 12V; or you can provide a special 14V supply for them.

If you want to provide a 14V supply, and you also have a 24V supply in your system, the easiest way is to use exactly the same procedure described above in 6.3V step-down converter. The only differences are that (1) you use the 24V supply as the input to this second converter, and (2) you'll dial in 14V when it comes time to adjust the output voltage.

If you don't have a 24V supply available, there's another alternative: you can use a "step-up" converter to convert 12V from your ATX power supply into 14V. This is almost exactly the same procedure as using a step-down converter, but in this case you have to specifically search for a "DC-to-DC **step-up** converter". A step-up converter has the ability to increase the input voltage, whereas a step-down converter can only limit the input to a selected lower voltage. Step-up converters are slightly more expensive, so the 24V step-down option is cheaper if you already have a 24V supply.

30-50V supply for replay knocker (and other pinball coils)

If you're using a real pinball replay knocker, it's probably designed to operate on 50V. This is the case if you bought a new knocker assembly from a pinball parts vendor; if you have an older knocker salvaged from a machine made before the 1990s, it might have a lower-voltage coil.

Similarly, if you're using real pinball assemblies for your flippers, slingshots, or pop bumper effects, and they're for machines from the 1990s or later, the coils in those are also designed for 50V operation.

Many pin cab builders run their 50V knocker coils using 24V power supplies, since that's the highest voltage supply that most cab builders install. This will work, but the effect will be weaker than it should be. To get the full effect like in a real machine, you need a higher voltage. You don't necessarily need the full 50V, but the closer you get, the stronger and more realistic the effect will be.

There are two straightforward ways to get a supply voltage closer to 50V.

Warning: 50V is a hazardous high voltage, so use appropriate caution if you install any type of supply in this range.

Add a 48V power supply: The easiest option is to buy a dedicated 48V power supply. You can buy OEM power supplies that produce this voltage. See OEM power supplies earlier in this chapter for instructions on buying and installing these.

Use a step-up voltage converter: This is almost exactly like setting up a step-down voltage converter, as described in 6.3V step-down converter. Follow the instructions in the 6.3V converter section, except that when you search for the part on eBay, look for a "DC-to-DC **step-up** converter". The "step-up" part is key,

because you need a converter that can convert from 24V to a higher voltage.

Which option is better? It depends on your setup, since each solution has pros and cons. A dedicated 48V supply is easier to set up and will have much higher power limits (Amps/Watts). But a step-up converter is cheaper and takes up less space.

My recommendation: if the only thing you're going to connect to the high voltage supply is a replay knocker, shop for each type, and use whatever's cheaper. If you have multiple high-voltage pinball coils that will share the supply, forget the converter and go with a dedicated 48V supply. It's too difficult to find a step-up converter with enough power capacity for multiple devices.

Variable supply for shaker motor

Most shaker motors nominally run on 12V, but some people find that the shaking effect is too strong if they use the full voltage. You can moderate the effect, if you wish, by reducing the voltage.

Note that if you're using a MOSFET-based output controller to control your shaker motor, such as a Pinscape power board or one of Zeb's booster boards, there's no need to adjust the voltage. You can adjust the strength via software instead. See Adjusting shaker strength via DOF below for how to do this.

If you're using a relay to control your shaker, though, the only way to adjust the strength is to adjust the power supply voltage to the motor.

The easiest way to do this is to use a variable step-down voltage converter, as described in 6.3V step-down converter. Follow the same procedure described in that section, but in this case, connect the converter output to the shaker motor's power input rather than to button lights.

In addition, you're not looking for a specific voltage in this case. You're only looking to adjust the shaking effect to your liking. So instead of using a voltmeter to dial in a specific voltage, you need to find the right setting by experimentation. Start by setting the output to the full 12V (using the voltmeter as in the 6.3V setup instructions). Run the shaker and observe the effect. If it's too strong, adjust the voltage downwards and try running the shaker again. If it's still too strong, turn it down some more; if it's too weak (or the shaker won't start at all), turn the voltage up. Repeat until you get the effect you want.

Adjusting shaker strength via DOF

If you're using a relay to control your shaker motor, the only way to control its strength is by adjusting the supply voltage. If you're using a MOSFET-based output control, though, such as a Pinscape power board or one of Zeb's booster boards, you don't need any voltage adjustment hardware. You can adjust the strength in DOF instead.

If you're already set up DOF, here's how to adjust the shaker strength in software:

- Open the DOF Config Tool
- Log in
- Click the Port Assignments tab
- In the "Device" drop-down, select the output controller device where your shaker motor is attached
- On the right side of the page, look for the section near the top labeled "Shaker Motor"

- Adjust the "Max Intensity" setting
- Click Save Config
- Click Generate Config
- Unzip the downloaded config files into your DOF folder
- Try running the shaker motor via a DOF test table in VP
- If the shaker effect is still too strong, go back to the config tool and decrease the Shaker Motor Max Intensity setting; if it's too weak, increase the setting

See Chapter 46, DOF Setup for more details on setting up DOF, using the Config Tool, and using DOF test tables in VP.

46. DOF Setup

One of the best tricks you do with a virtual pin cab is feedback effects: things like flashing lights, shaker motors, fans, solenoids, replay knockers, and bells and chimes.

To connect output devices to your pin cab PC, you need a special hardware device called an output controller. That's described in detail in Chapter 12, I/O Controllers. Once you have an output controller installed, you have to set up some special software called "DOF", or DirectOutput Framework, that serves as the middle-man between the output controller and the pinball software (such as Visual Pinball). This chapter is all about DOF and how to set it up.

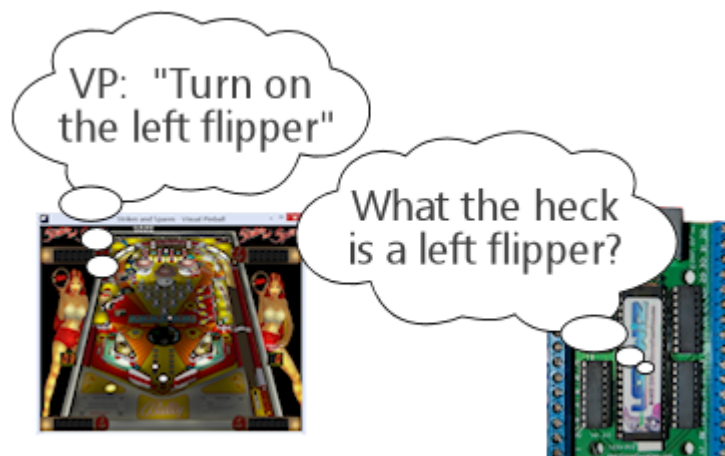
If you're not going to include any feedback devices in your pin cab, you won't need to install DOF at all, so you can skip this section.

What is DOF?

DOF is a separate piece of software that you can install on your system to control feedback devices. It's not really a full separate program; it's more of an add-on that works together with Visual Pinball and other software. DOF handles the communications between the pinball software and your output controller devices. It performs two main functions.

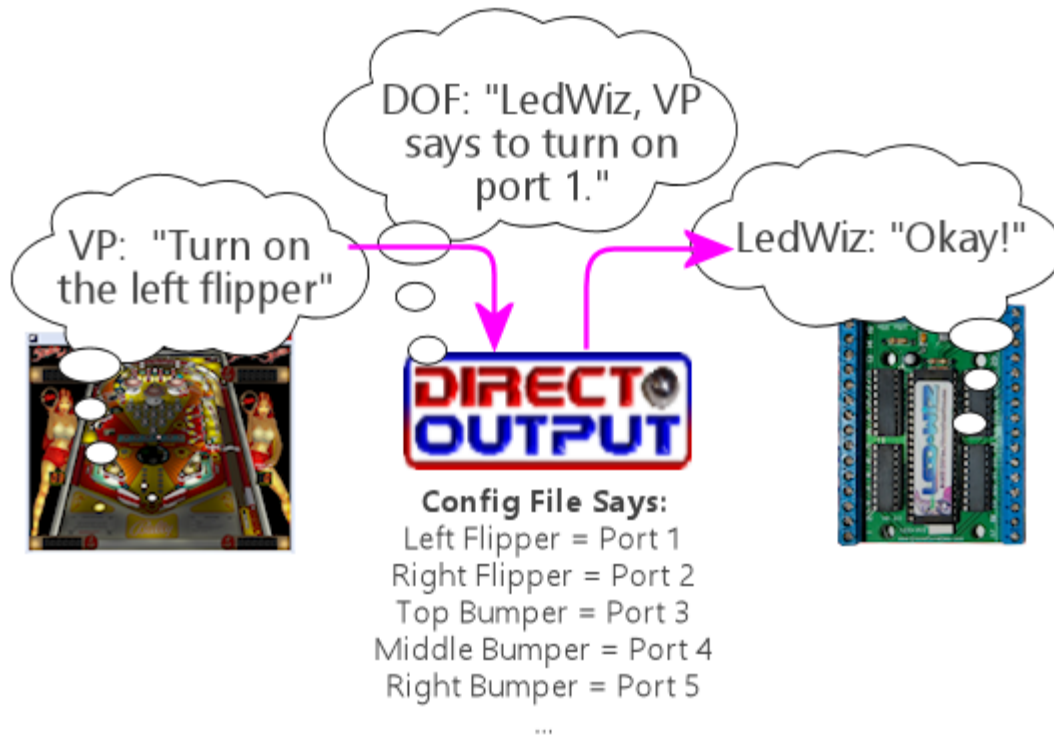
First, DOF handles communications with several different kinds of output controllers: LedWiz, PacLed, SainSmart relay boards, Pinscape, and more. Before DOF came along, a program that wanted to access multiple output controllers had to be specially programmed with each controller's language. DOF provides a common interface that VP can use to access any of these devices without having to know its details.

Second, DOF translates between VP's simulated game elements and the output controller's physical device connections. VP thinks in terms of the elements in the simulated pinball table: left flipper, middle bumper, right slingshot, etc. The output controllers, on the other hand, don't know anything about pinball. They only know about generic devices attached to their numbered "output ports": port 1, port 2, port 3, etc. Each output port on an output controller is basically a wired connection to a physical device such as a solenoid or flasher light. But the output controller devices don't know anything about what's attached; they just know there's *something* wired to each port.



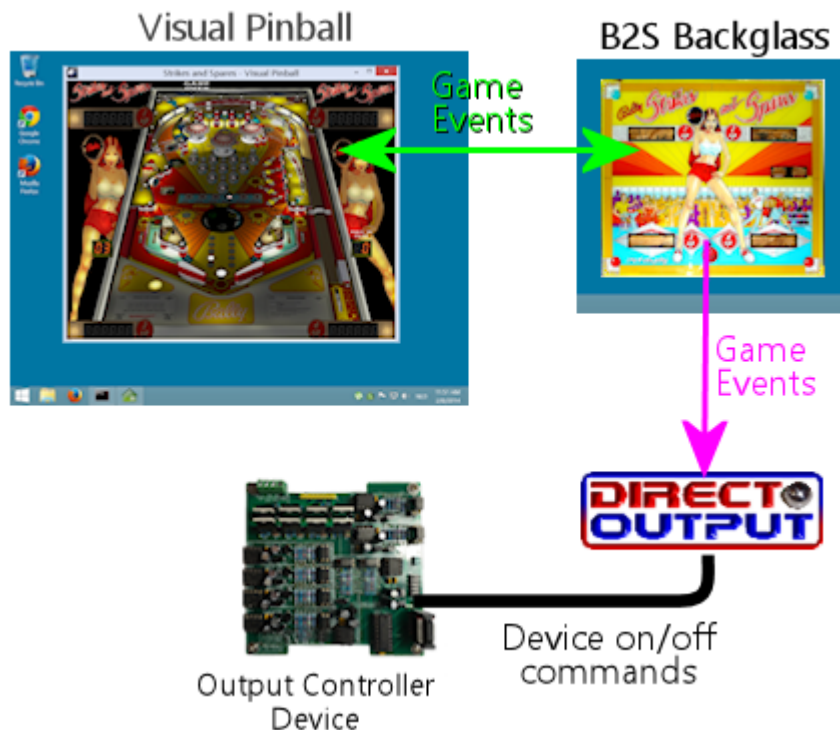
DOF acts as an interpreter to translate between VP's game-oriented model and the

output controller's physical device list. DOF accomplishes this using a configuration file that you provide, which tells it the "meaning" of each physical output port.



How VP and DOF communicate

DOF doesn't actually talk directly to VP. It talks to VP through the B2S Backglass program, which is another VP add-on that handles the backglass artwork display.



The first step is that VP communicates with the B2S. VP and B2S exchange information on game events, which are things like "the ball just hit switch #22" or "lamp #17 just turned on". B2S in turn sends the game event information to DOF. DOF translates the game events into feedback device actions, and sends the appropriate ON and OFF commands to the output controller device.

How this relates to individual tables

Newcomers to DOF often have a question at this point: Is this already programmed into every VP table?

The answer is a qualified Yes. Almost every VP X table has full DOF support built in. Most later VP 9 tables also have DOF support, although many VP 9 tables require you to make a small manual change to their scripts to activate DOF support. We'll explain how to do this later.

How to install DOF

Before you install DOF, you must first install Visual Pinball and the B2S Backglass program. If you haven't already installed those, go back to Chapter 15, Pinball Software Setup and follow the instructions there. If you're planning to use PinballX as your game selection menu system, you should also install that first.

Once VP, B2S, and PinballX are installed, you're ready to install DOF. You have two options for this: automatic or manual.

In either case, you'll have to choose a folder for the DOF files. I recommend **C:\DirectOutput**, if that's convenient on your system. You can use a different name if you wish, but note the following:

- **Avoid using spaces in the DOF folder name.** A few people have run into problems that were very difficult to isolate, but which seemed to come from using a folder name containing a space character, such as "C:\Direct Output". Even though this *shouldn't* be a problem, it does seem to cause weird errors on a few machines, so I'd avoid it.
- **Try a C: folder if it doesn't work on another drive.** Some people have had problems installing DOF on other drives (like D: or E:). As with the space character issue, it *should* be just fine to use D: or E: or any other drive, but doing so seems to cause problems on some machines. There are sometimes good reasons to use a disk other than C:, so go ahead and try it if you prefer to keep your programs on a different drive. But if you run into any mysterious problems, try moving it to the C: drive to see if that helps.

System requirements

If you're on Windows 7, make sure your system has **.Net 4.6** installed before proceeding. That's a core Windows component that DOF requires. It can be installed via Windows Update. Most newer systems will already have this installed, but you might have to install it manually if you're using Windows 7 and recently did a fresh install of the operating system.

Automatic DOF Setup

The easy and recommended way to install DOF is to use the Windows Setup-based automated DOF installer. You can find the automated installer here:

Grander Unified DOF R3++

Look for the latest "Windows Setup (MSI)" edition. Download it and run it. When the installer asks you to pick a folder name, remember to use a name that doesn't contain any spaces.

If all goes well with the automatic setup, you can skip all of the "manual setup" steps listed below. The installer takes care of everything for you. Skip directly to Check that it's working below.

Troubleshooting: The automatic setup is usually pretty reliable, but there are a couple of issues that sometimes come up. If you run into any error messages that aren't self-explanatory, try these steps:

- Check that UAC is **enabled**. Some people disable UAC based on advice they find on the Internet. That kind of advice is all badly out-of-date (from the Windows Vista days), but it keeps getting repeated anyway. It's bad advice these days. UAC is an integral part of modern Windows. Disabling it changes some system internals in ways that create headaches.
- Mysterious permission errors might be due to a special system "owner" of the C:\ root folder, or wherever you're trying to install DOF. You can check the owner as follows:
 - Right-click the folder on the desktop
 - Select Properties from the Context menu
 - Go to the Security tab
 - Click Advanced
 - Look for the "Owner" listing

If the owner is SYSTEM or Trusted Installer, the root folder is protected against writing by ordinary user accounts. You can either choose a different parent folder that's not protected like that, or you can run the installer in "Run as Administrator" mode. In the latter case, the new folder will also have a protected system owner when you're done, which will create other problems later. But you can fix that after the install completes, by changing the new folder's owner to match your regular user account:

- Repeat the process above to get to the Owner for C:\DirectOutput
- Click the Change link next to the owner name
- Type your regular user account name in the box
- Click Check Names
- Make sure this expands to a valid Domain\Username and click OK

Manual DOF Setup

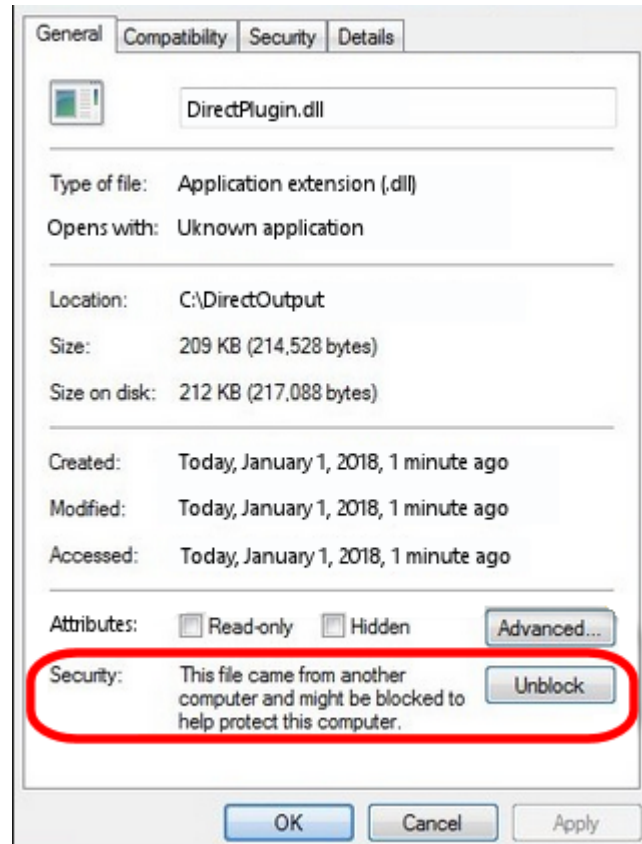
I strongly recommend using the automated installer above rather than attempting a manual installation. DOF is notoriously difficult to install by hand; it has a million fiddly little details that you have to get right, and any small oversight breaks the whole thing. And DOF is bad at explaining what's wrong when something does go wrong, so it's extremely difficult to troubleshoot bad installs. The automated installer has proven to be much more reliable.

If you insist on doing it the hard way, though, here's my recommended manual setup procedure.

Note! To keep things as simple as possible, the instructions below leave out some details that most people don't need. If you want the full story, see the DOF documentation, which you can find via the DOF documentation links later in this chapter. In addition, the DOF version we link below isn't the only one available. There are some other modified versions with slightly different extra features available. See release status for details.

- Create a DirectOutput folder on your PC called **C:\DirectOutput** (or a name of your own choosing, but remember that it must not contain any spaces)
- Download the **ZIP file edition** of my Grandeur Unified DOF R3+

- Unzip the contents into your new DirectOutput folder
- Unblock all of the new DLL and EXE files. For each file in the new folder with a **.dll** or **.exe** suffix, do the following:
 - Right-click the file
 - Select Properties from the menu
 - Select the "General" tab in the properties window
 - Look for a message like this: "Security: This file came from another computer and might be blocked to help protect this computer"
 - If you see the message, click the **Unblock** button next to it



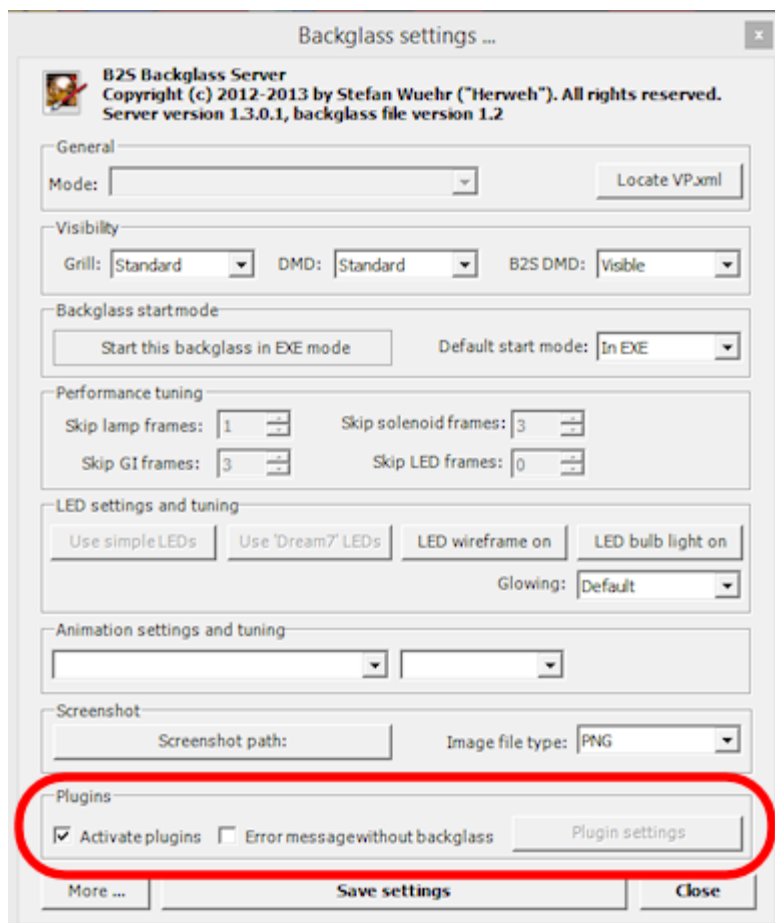
- Open the folder where you installed Visual Pinball. Open the sub-folder **Tables**. Look for a sub-folder called **Plugins**. If it's already there, great, otherwise create it:
 - Right-click in a blank area within the **Tables** folder window
 - Select **New > Folder** from the menu
 - Type **Plugins** to set the new folder's name
- Open the **Plugins** folder that you just found or created, then:
 - Right-click in a blank area in the folder window
 - Click **New > Shortcut** in the context menu
 - Type the full path to your Direct Output folder into the box (e.g., **C:\DirectOutput** - this is the folder you created above, at the very beginning of this process)
 - Click the Next button
 - Type **DirectOutput** for the name
 - Click the Finish button

- For the next step, you'll need a Visual Pinball table **that includes a B2S backglass** installed. The backglass has to be in a separate file with the same name as the **.vpx** table file, but with a **.directb2s** suffix. If you don't have any of these table/backglass pairs installed already, you'll have to install one now. For this test, you can use **2001**, because it's easy to find, being one of the first ones in the list on vpforums.org:
 - Open vpforums.org in your Web browser
 - Log in (create an account there if you don't have one yet)
 - In the navigation bar near the top of the main page, click "Visual Pinball Tables"
 - In the box that pops up, look for the "VPX Tables" section, and click "All"
 - Click on "2001 (Gottlieb 1971)", which should be near the top of the list (if not, try any of the other tables)
 - Click the "Download" link and follow the instructions to download
 - Unzip the downloaded file into the **Tables** folder inside your Visual Pinball program folder
- Make sure that the ZIP file you just downloaded included both a **.vpx** file and a matching **.directb2s** file. If not, you'll have to try downloading other tables until you find one that includes both, because the backglass file is required for the next step. Alternatively, you can look for the matching **.directb2s** file separately:
 - Click **Frontend Media & Backglass** on the [vpforums](http://vpforums.org) navigation bar
 - Click **dB2S Animated Backglasses** in the popup box
 - Search the list for the matching file
 - Click on the file and download it as above
 - Make sure the downloaded file has the **same filename** as the **.vpx** file for your table, with **.vpx** replaced by **.directb2s**. You can simply rename the B2S file manually if its name isn't an exact match.
- Once you have a VP 10 table and matching backglass ready to try, load it into VP 10 and run it. This should display the table and backglass in separate windows.
- Right-click anywhere on the backglass. This should bring up the B2S options dialog. It should look like this:



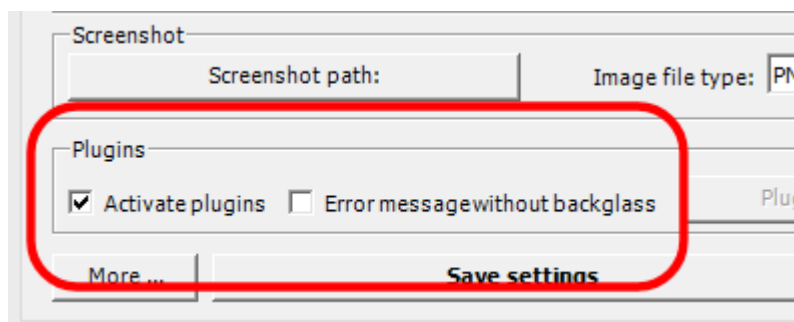


To bring up the B2S options dialog, you have to run a VP 10 table that has a matching B2S backglass file installed. Running the table from within Visual Pinball will display the table and backglass in separate areas on your screen. Right-click the mouse anywhere in the backglass area to bring up the B2S options dialog.



The B2S options dialog. The "Plugins" section at the bottom is what we're interested in here.

- Check the box next to **Activate plugins**, and un-check the box next to **Error message without backglass**.



Make sure that **Activate plugins** is checked, and **Error message without backglass** is un-checked

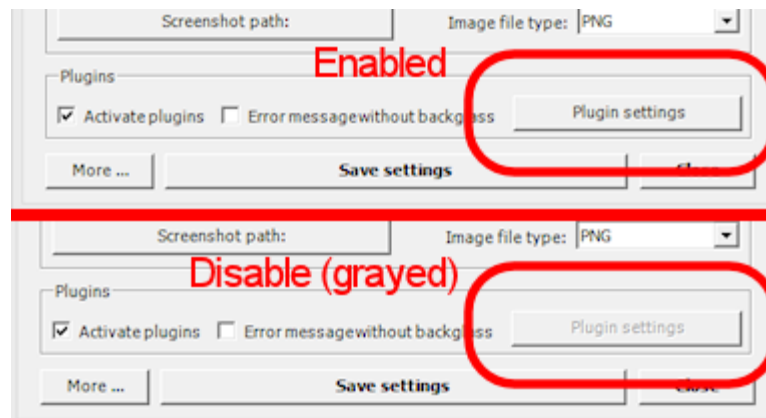
- Click Save Settings
- Exit the table (press "Q" and then "Q" again) and close VP

Check that it's working

Before proceeding, make sure you close all VP windows that you might have had open from the steps above. You want to make sure VP has a chance to restart with the new settings.

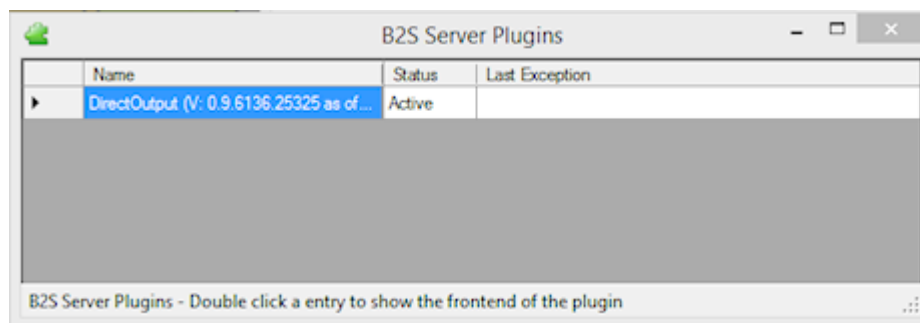
Now start VP, and load a table that has a B2S backglass. You can use the same table you used during the setup procedure in the step where we updated the B2S backglass settings.

As before, when the backglass appears, right-click the mouse anywhere in the backglass display area to bring up the options dialog. Look to see if the **Plugins** button at the bottom is enabled:



If the button is disabled, DOF isn't getting loaded. Go to the troubleshooting section below for things to try.

If the button is enabled, click it. This will bring up a separate dialog that shows the status of each plugin.



Look for a **DirectOutput** entry in the list. If you don't see any such entry, it means the same thing as a disabled Plugin Status button, namely that DOF isn't being loaded. Go to the troubleshooting section for help.

Finally, check the **Status** and **Last Exception** columns for the Direct Output entry.

If the Status is Disabled, or there's a message in the Last Exception box, see the troubleshooting section for help.

If the Status is **Active** and the Last Exception column is empty, congratulations! Your DOF setup work was successful! DOF is loading and starting correctly.

Extra controller setup

If you have any of the following controller types, you have to do some additional work to tell DOF how to access them:

- Chapter 51, SainSmart USB relay board
- Chapter 65, Teensy addressable LED strip controller

If you're not using one of the controllers listed above, you can skip to the next section. Most other controller types **don't** require any extra configuration work on your part, because DOF finds them automatically each time it runs. DOF automatically detects Pinscape, LedWiz, and Pac-Led.

If you're using one of the controllers that requires extra configuration, follow these steps:

- In your DirectOutput folder, check for a **config** folder. If it's not already there, create a new folder and name it **config**.
- If you're using my DOF R3++ version, there should be an **examples** folder inside the **config** folder. Go to that folder and copy the files there to the **config** folder. If there's no **examples** folder, download the following files into your **config** folder (these are the same files included in my DOF R3++ version):
 - mjrnet.org/pinscape/downloads/DOFConfigSamples/GlobalConfig_B2SServer.xml
 - mjrnet.org/pinscape/downloads/DOFConfigSamples/Cabinet.xml
- In your DirectOutput folder, run the program file **GlobalConfigEditor.exe** by double-clicking it
- On the menu at the top of the window, select **File > Load**
- Navigate to your **DirectOutput > config** folder and select **GlobalConfig_B2SServer.xml**
- Click on the Cabinet Config tab at the top
- Click Select File
- Navigate to your DirectOutput > config folder and select **Cabinet.xml**

DOF should now load Cabinet.xml every time you start a game in Visual Pinball. The Cabinet.xml file provided above is just a starting point, though - you still have to edit it to add information on your Sainsmart relay board or Teensy light strip controller. See the sections for those devices for details on what to add to the file.

The DOF config tool

The next (and nearly final) step is to tell DOF how your feedback devices are connected to your output controller. If you haven't already started installing your feedback devices, you might want to skip this section for now and come back to it when you get to that point.

The basic thing we have to do is tell DOF which type of device is connected to each port number on your output controller.

What's a "port number"? Every output controller is a little different, but they all give you a set of wiring terminals where you connect your output devices. For example, the LedWiz gives you two rows of screw terminals that look like this:





You connect one device to each screw terminal. For the details of how the wiring is actually connected, see Chapter 47, Feedback Device Wiring, but for our purposes here, let's just think of it like this: each device is connected to one terminal on an output controller.

You'll notice that there's a number printed next to each terminal on the LedWiz board. Those are the port numbers we mentioned. Every physical wiring terminal has a port number assigned.

You'll also notice that there's *not* anything printed on the LedWiz about "Left Flipper", "Shaker Motor", "RGB Flasher #1", or anything like that. So which terminal are you supposed to connect the shaker motor to? The answer is easy: it's up to you, so just pick one! As far as the LedWiz (or other controller) is concerned, all of the ports are the same. They're just general-purpose outputs that you can connect to just about anything. The LedWiz doesn't have to know anything about what's connected, because its only job is to turn the port on and off when commanded by the software.

But if the ports are all the same, how is DOF supposed to know which port is the shaker motor, which port is the left flipper, and so on?

That's where the DOF Config Tool comes in. The Config Tool lets you set up exactly this connection between port numbers and device types. Which is why we said earlier that you should have already mapped out your device wiring before you get into this step. You need to be able to tell the Config Tool which device you're going to attach to which port number, so you'll need at least a plan for how your ports are laid out.

Step 1: Log in

The DOF Config Tool is an online tool that you access from a Web browser. Here's the link:

configtool.vpuniverse.com

If this is your first time here, click "Create Account" in the top navigation bar to set up a new account. An account is required because the Config Tool has to store each user's unique cabinet setup data separately.

Step 2: Select your output controllers

After creating an account, the next step is to click "My Account" on the navigation bar. This takes you to a page where you can tell the tool which output devices you have.

Go through the list and tell the tool which devices you have. If you have only one type of controller, all you have to do is find that type in the list and set its drop-down to "1". The number simply indicates **how many units** you have of each type, so if you have a single unit, set it to "1".

If you're using a Pinscape controller, set **Number of Pinscape devices** to 1 and leave "Number of FRDM-KL25Z Devices" set to 0. This is a little confusing: Pinscape runs on a KL25Z, so it might seem like, technically, you do have one of those as well.

But the "FRDM-KL25Z" listing in the Config Tool really should be labeled "old Pinscape v1 firmware". Assuming you're planning to use the modern Pinscape firmware, just say that you have one Pinscape unit and zero KL25Z's.

When finish setting the output controller selections, click "Save Settings" to save the updates. Note that, throughout the Config Tool, you always have to click the Save button before leaving the current page if you want changes to stick. If you navigate away from a page before saving, any changes made on that page are usually discarded.

Note that your settings in the Config Tool are never set in stone. You can always come back to this page later to make changes, if you ever add a new output controller, for example, or change to a different one.

Step 3: Set up your port assignments

After saving, click "Port Assignments" in the nav bar. This will take you to the page where set up the mappings between output port numbers and specific devices. We've finally reached the point where we're talking about concrete, specific devices!

This page lets you work on one output controller at a time. If you have more than one controller in your system, you simply set up each one separately. The "Device" drop-down at the top of the page lets you select the one you want to work on. As always, remember to save any changes before selecting a different device.

For now, ignore the boxed items on the right side of the page ("Shaker Motor - Min Intensity - Max Intensity", etc). These are for fine-tuning your setup once you have everything working. It's best to leave the defaults in place initially.

During this step, we're going to set up the "Port *number*" items on the left side.

The number of "Port" items shown on the page depends on the type of output controller you're using. For an LedWiz, for example, there should be 32 ports, Port 1 through Port 32, because that's how many physical ports an LedWiz has.

For a Pinscape controller, the page will show 128 ports. You might not have that many physical ports in your setup, but that's the maximum that the firmware can handle. Your actual number of ports depends on how whether or not you're using the expansion boards, and if so, which ones you're using and how many of each. To keep things simple, though, the Config Tool ignores all of that and just gives you the theoretical maximum of 128 slots. You should simply treat any slots beyond the ones in your actual system as "reserved for future expansion", in case you add more expansion boards later, for example. Just leave any extra slots blank on the Config Tool page.

To set up the port mappings, all you need to do is go through the ports one by one, and select the device type attached to each port from its drop-down list. If you've already connected your feedback devices to their output ports, hopefully you kept notes on which device was wired to which port! Get out those notes now and enter the same information the Config Tool port list.

If you're not sure what any of the terms in the drop-down list mean, see Appendix 5, DOF Config Tool Device Descriptions. That provides a full list of all of the devices in the drop-down lists, with detailed explanations of how they're usually implemented in virtual cabs. The devices in the drop lists are mostly self-explanatory, but some of them are pretty obscure, plus there's a certain amount of "virtual pinball jargon" that probably won't make much sense if you haven't spent a lot of time in the forums.

Once you enter all of the devices, click Save.

Step 4: Generate your config files

If you're still on the Port Assignments page, you should see a button near the top called Generate Config. Click it. The Config Tool will now create your customized configuration files and download them to your PC as a ZIP file.

Wait for the download to complete. Open the ZIP file. Unzip the contents into your C:\DirectOutput folder (or wherever you installed the DOF files back at the start of this process).

Important: Unpack **all** of the .ini files from the ZIP file generated by the Config Tool, and **don't rename any of them**. Some people get confused by the multiple files and think you're supposed to choose one of them. You're not. You need **all** of them, with the exact names generated. Each file corresponds to one output controller, and the number in the name (if any) tells DOF which controller the file goes with. If you don't unpack all of them, or if you rename any of them, DOF won't work correctly.

Step 5: Test it

You should now have a fully working DOF setup! You should try running a DOF testing table to check that the commands can make it all the way through from Visual Pinball to your devices. See testing below for instructions.

Update your config any time you change your device setup

Any time you change anything in your cabinet that affects the DOF setup, you'll have to return to the Config Tool, make appropriate changes to the settings there, and then re-generate your config files. The Config Tool remembers your saved settings between sessions (that's why you have to create a user ID and log in), so you'll only have to enter any new or changed information for your output controller list or your port assignments. After you make any needed changes, repeat the Generate Config step: click the button, download the ZIP file, and unzip the contents into your Direct Output install folder. Simply replace the old copies of the config files each time you do this.

Running a DOF test table

The final and most important test is to see if Visual Pinball can successfully control your feedback devices during a game. The easiest way to do this is with a VP table specially designed for testing DOF.

Here's a good test table you can use:

DOF Test Table VPX

(If that link doesn't work, try searching the vpuniverse files section for "DOF Test Table VPX", or just do a Web search for the same term.)

Once you find a suitable test table, download it, unzip it into your **Visual Pinball > Tables** folder. Open it in VP and press F5 to play the game. The test table linked above provides on-screen instructions with keys to press to try activating different devices. You can go through your attached devices and verify that they work.

If none of your devices work, your DOF setup probably has a configuration problem. See the troubleshooting section below for help.

If some of your devices work and some don't, you can be certain that DOF itself is working, since a software problem with DOF would prevent anything from working.

Check the wiring to the non-working devices, and double-check that they're set up correctly in the DOF Config Tool. For example, double-check that the port number that you entered for each device in the Config Tool matches the port number printed on the controller board where the wiring to the device is connected.

How to enable DOF in VP

There are two requirements for a game to work with DOF:

- First, it has to be listed in the DOF Config Tool's database. To see the current list of supported tables, log in to the Config Tool, go to the Table Configs tab, and open the Table Name drop-down list. This has all of the tables that the Config Tool currently supports.
- Second, the table's script in Visual Pinball has to include B2S backglass support.

With VP 10, almost every table automatically uses B2S if it's installed. There's usually nothing you have to do as a player to enable DOF for these tables; it should just work when you run the table.

With VP 9, the situation isn't nearly as DOF-friendly. Most tables in VP 9 require a small amount of script editing to enable B2S support, which in turn enables DOF support.

Both DOF and B2S came onto the scene during the years that VP 9 was the dominant version, so there was a lot of evolution of the common scripting practices among VP 9 table authors. As a result, there's not a simple formula for "fixing" VP 9 tables to use B2S and DOF. Many later VP 9 tables, written around 2016, have native B2S support, and you won't have to do anything to get it working. Slightly earlier tables have support for B2S coded in and ready, but disabled by default; you have to do a little script editing to enable it. And most earlier tables have no pre-coded support for B2S, but support can be added with some minimal script editing.

To determine which type of VP 9 table you're working with, start by downloading the table file (.vpt) and the matching backglass file (.directb2s). Make sure that both files have the same name, except for the respective .vpt and .directb2s suffixes. Now:

- Open the table in VP 9
- On the menu, select **Edit > Script**
- Look for a line like this:

```
Const cController = 0 ' 1=VPinMAME,  
                     ' 2=UVP backglass server,  
                     ' 3=B2S backglass server
```

- If you find such a line, simply change the "0" to the number listed for B2S. (Or, if the comments also call out a setting specifically for DOF, use that number instead.) Save and run the table. If it successfully displays the B2S backglass, you're set.
- If you can't find a "cController" line like the one shown above, try searching for a line that looks like this:

```
Set Controller = CreateObject("VPinMAME.Controller")
```

- When you find that, replace it with this:

```
Set Controller = CreateObject("B2S.Server")
```


- Save, and try running the game. Again, if it displays the B2S backglass, the table should now work with DOF.
- If you can't find any mention of "VPinMAME.Controller" anywhere, it's probably not a ROM-based game. In this case, the table *can* be converted to use DOF, but it requires substantial custom scripting work that's beyond the scope of this chapter.

Disabling unwanted sound effects in a VP table

VP tables are mostly written with desktop play in mind, so they assume that you want digitized sound effects for every mechanical event in the game: flippers, slingshots, bumpers, replay knockers, gear motors.

The whole point of using DOF feedback devices is that real mechanical devices produce more realistic audio and tactile effects than recordings. But when you play a DOF-enabled table on a DOF-enabled cabinet, you'll notice that VP often still plays those canned sound effects, on top of the real mechanical action that DOF is providing.

If you're like most DOF users, you'll probably find that recorded sound effects sound artificial and redundant when played at the same time as real mechanical DOF effects. So you'll probably want to turn off the canned effects that match up with the toys you have installed. If you have flipper solenoids installed, for example, you'll probably want to turn off the simulated flipper sounds VP plays via audio.

Fortunately, you can do this. VP 10 makes it easy. At least, it makes it easy for properly programmed tables.

- Open VP X in "editing" mode (no table needs to be loaded)
- On the menu, select **Preferences > Keys, Nudge and DOF**
- Look for the **DOF Controller Options** section on the right side of the dialog
- Go through the listed devices. For each one where you have a DOF device installed, change the setting from **Both** to **DOF**.
 - **Contactors** refers to the bumpers and slingshots
 - **Knocker** is the replay knocker
 - **Chimes** refers to EM-style chimes
 - **Bell** refers to the bell used in some games
 - **Gear** refers to the gear motor
 - **Shaker** refers to the shaker motor
 - **Flippers** refers to the flipper solenoids
 - **Targets** controls sound effects produced when the ball hits stand-up targets; with DOF, these use the bumper devices to simulate an impact effect
 - **Drop Targets** controls sound effects produced when the ball hits drop targets; with DOF, these use the bumper devices

These settings will only work with VP 10 tables that were scripted properly. Most newer tables should conform, but you might find a few that don't, which will be noticeable because they won't properly disable the sound effects according to your settings changes above. You might be able to fix such a table by following the procedures for VP 9 tables below. Or you can just contact the table's author and suggest updating the table to use the modern scripting conventions that take DOF

into account.

In VP 9, the same thing is possible, but unfortunately, it's not nearly as easy. VP 9 doesn't have option settings for the individual DOF toys. What you have to do instead is edit the individual table script for each table you want to fix.

In older tables, you usually have to edit the scripts by painstakingly scanning through the scripts, finding all of the sound effects commands, and removing the ones you don't like.

Many later VP 9 tables (written in 2015 and later) include pre-programmed support for removing individual sounds, but it still requires you to edit the script to activate it.

Here's the basic procedure for both kinds of tables:

- Open the table in VP 9
- On the menu, select **Edit > Script**
- Look through the comments at the top for options relating to "Sound Effects", "DOF Sounds", or "Cabinet Sound Options". Some scripts offer variables to set DOF mode (which usually turns off all canned mechanical sounds), and some have several variables to selectively turn off certain mechanical sounds.
- If you don't see any such comments, you can still disable selected sounds yourself by manually editing the script, but it will take a lot of work:
 - Search for **PlaySound**. The name in quotes after **PlaySound** is the sound effect file to play; this usually has a name that suggests its purpose. Each time you find a PlaySound line that has a sound you want to turn off, you can "comment out" the line by putting an apostrophe (') at the very start of the line.
 - Search for **vpmSolSound**. This is another way that scripts play sound effects. For these, don't comment out the line, but instead delete the name inside the quote marks that follow, leaving the quotes intact. For example, replace "**vmpISolSound** **""Knocker""**," with "**vmpISolSound** **""""**,".

Full documentation links

The official DOF R3 documentation is here: pinball.weilenmann.net/docu/DirectOutputWIP/index.html

You can also find the documentation on the DOF project page on GitHub: directoutput.github.io/DirectOutput/. However, as of this writing, that version has only been updated as far as the older R2 version.

DOF release status

DOF's release status is a little confusing, because its original author, SwissLizard, suspended work on the project before finishing a major update that was in progress. He released a few "beta" test builds of the new "R3" release in late 2015, but he never completed the official, final R3 release. Fortunately, he published the project under an open-source license, so other people have been able to continue work on the project. It's very much alive and well as a result. The downside is that multiple unofficial versions have emerged. That creates a little more work for you, since you have to decide which one to use.

Here are the main options:

- My latest "Grander Unified" DOF R3++ is a merge of all four of the forks of DOF on GitHub as of January 2018, plus some additional updates I've made since then. As of this writing, it has every feature of every alternative version, so you don't have to choose among versions with subsets of features. This is the one I recommend because it has everything all of the other versions offer.
- My original "Grand Unified" DOF R3+ incorporated all of Swiss Lizard's final published code, plus some necessary changes to support Pinscape devices. This also contains my LedWiz enhancements.
- Swiss Lizard's own beta R3 releases are badly out of date at this point, but if you want to find them anyway, try this Web search: `site:vpforums.org DOF R3 beta`. That should turn up the DOF R3 beta announcement thread on the forums, which contains links to the download files. Look towards the end of the thread for the newest updates.
- The last "official" (non-beta test) release from Swiss Lizard himself was the R2 version. This is even more out of date than the R3 betas, but if for some reason you want something that's nominally official, this is it. Go to vpforums, click on "Getting Started" on the top navigation bar, then select "Frontends and Addons" from the "Essential Files" section of the popup menu. This will take you to a list of files. Find "DirectOutput Framework R2" in the list. Click on the link. This will take you to a download page. (If Swiss Lizard ever officially releases a final R3 build, it should also appear here in due course.)
- Three other developers (as of January 2018) have created their own forked versions of the R3 code on GitHub, to add their own extensions. Check the main DOF page on GitHub to see the current list of forks. All of the forks that were active up through January 2018 were merged into my Grander Unified R3++ mentioned above, and I'm not currently aware of any further work on any of those forks beyond what I merged.

My intention with the Grander Unified R3++ version of January 2018 is provide a One True DOF, re-unifying all of the forks and eliminating any confusion about which one to use. The unified version combines all of the features from all of the forks, so there's no need to pick a subset of features.

Troubleshooting your DOF setup

When DOF doesn't just work the first time you try to set it up, it can be a real pain to figure out why. The big problem is that DOF doesn't have very good error reporting. When something goes wrong, DOF often gives no indication what the problem might be, leaving you to make wild guesses until you hit upon the solution.

Taking shots in the dark is an inefficient and time-consuming way to debug a problem, and it can often make things worse if you try random changes without thinking things through. So the first thing you should do is not panic. Don't try random things. Instead, follow the steps below. DOF does offer a few subtle clues about the nature of the problem when something goes wrong, if you know where to look. We'll try to help you read the tea leaves to figure out what's going wrong and how to fix it.

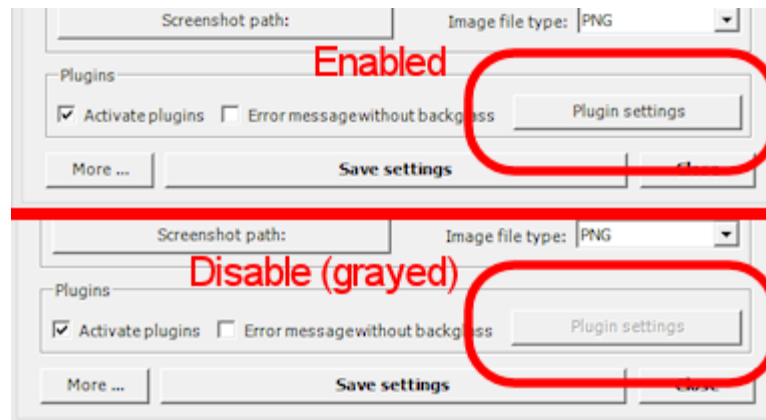
Step 0: Check your Windows configuration. Make sure that UAC (User Account Control) is **enabled**. Some people disable it because of old/bad advice on the forums. Don't. Turning off UAC doesn't do what people think; it changes some Windows internals in subtle ways, and some people have had DOF problems as a result.

Step 1: Make sure you only have one copy of B2S installed. For whatever reason, a lot of people have run into the bizarre situation where they have multiple copies of B2S installed on their system. This can send you down blind alleys for hours. You spend a lot of time trying to get the B2S configuration files right only to find that you're changing an old copy that's no longer in use.

So before wasting a lot of time, **search your entire hard disk** for the main B2S files and make sure you only have one copy installed. A good file to look for is **B2SBackglassServer.dll**.

If you do find extra copies, delete them. Then go to the correct folder - which should have the one remaining copy - and run **B2SBackglassServerRegisterApp.exe** by double-clicking the file. It's important to run this step because it updates some Windows Registry settings to point to this copy. If you had an old copy somewhere else, the Registry settings might have left been pointing to the (now deleted) old copy.

Step 2: Check the B2S Plugins button. Open the B2S settings dialog by running a VPX table and right-clicking the mouse in the backglass area. Check the status of the **Plugins** button.



If it's disabled, it means that the DOF .dll files aren't being loaded at all. In this case, don't even think about what might be wrong with your DOF config files or anything like that. You have a very basic problem where DOF isn't even getting into memory. Here are the main things that can cause this, and how you might be able to fix them:

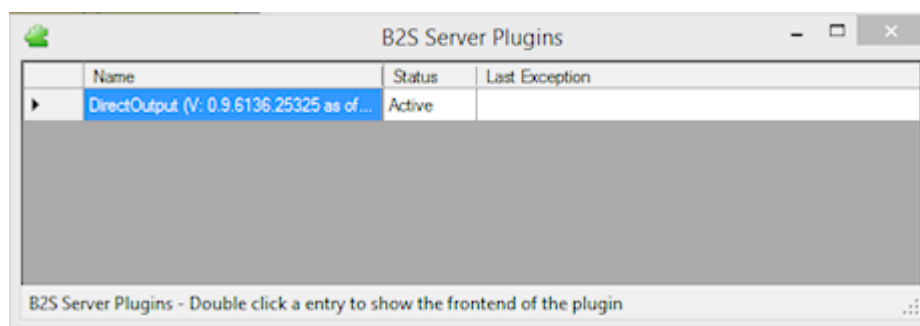
- The B2S backglass program doesn't know that you want it to load DOF in the first place. Make sure that the **Activate plugins** box is checked. If not, check it now, click Save Settings, quit out of the table, exit out of VP, and try checking again with a new VP session.
- B2S can't find the DOF .dll files. Check the **DirectOutput** shortcut in the **Visual Pinball > Tables > Plugins** directory. Make sure the shortcut points to the correct folder, the one where you installed the Direct Output .dll files. Make sure that everything is named correctly: the **Plugins** folder itself, the **DirectOutput** shortcut file, and the directory link within the shortcut file. The exact names are critical, so put on your proof-reading glasses and check carefully.
- B2S can find the .dll files, but it can't load them. Go back to the Direct Output folder and bring up the Properties window for each .dll file (right click on the file and select Properties from the context menu). Make absolutely sure that you've unblocked every file. Many people who have problems getting DOF to load find that the culprit was blocked .dll files, even though they were certain they unblocked everything the first time through. Double-and triple-check the

files, and make sure you look at every single one with a .dll suffix.

- Search your entire hard disk (using the Windows desktop search tools) for extra copies of Direct Output that you might have installed by accident or at different times. Some people have had this experience: they keep checking and re-checking the .dll files to make sure everything's unblocked, and it is. And after the umpteenth time, they realize they've been unblocking a second copy of the files in a whole separate location from the ones that B2S is trying to load.

If none of that helps, you might want to try deleting everything in your Direct Output folder and downloading a fresh copy of the files.

Step 3: Check the DOF plugin status. Click the **Plugin Status** button above (if it's disabled, go back to step 1). This should bring up a new dialog showing the status of each plugin.



Find the entry for **DirectOutput** in the list. If there is no DirectOutput entry, it means the same thing as a disabled **Plugin Status** button: B2S never managed to load DOF in the first place. Go back to step 1 above, because this is exactly the same problem described there with exactly the same possible causes.

If the DirectOutput entry is there, it means that DOF has been loaded. Now check the rest of the entry. If the **Status** column says **Active** and the **Last Exception** column is empty, it means that DOF was successfully loaded and started without any fatal errors and is running properly, at least as far as B2S is concerned.

If the status is **Disabled**, and an error message is displayed in the **Last Exception** box, it means that a fatal error occurred trying to load the DOF program files. Right-click on the Last Exception box and select **Show Exception Details** on the menu to see the full text of the exception. These messages are usually much too long to fit into the little box. Unfortunately, they're also much too technical to be of any help unless you're familiar with the inner workings of DOF's source code. Even so, it's worth taking a look at the message to see if there's anything in it that suggests to you what the problem might be. Don't feel bad if it looks like so much nonsense, though; B2S makes no attempt at all to interpret these internal system error codes into anything meaningful to humans.

So what do you do with these error codes that were never meant for you to see? I'm afraid there's not much you can do with them other than copy them into a forum posting asking for help. There are people on the forums who know the internals of B2S and DOF who can often help you track down the problem given these technical details, so you can try posting to see if someone can help you out. Post the full text of the exception message, along with details about your directory layout and anything you've already tried to fix the problem.

Step 4: Check the log file. If B2S says that DOF is loading properly, but it's not controlling your devices properly, you can check DOF's log files to see if there are

any errors there. If you tell it to, DOF will write a bunch of status information to a log file as it runs. This can be helpful when a problem occurs, since the status information sometimes has details about the specific cause.

First, make sure logging is enabled:

- Open your DirectOutput folder in Windows Explorer
- Run the program file GlobalConfigEditor.exe by double-clicking on it
- On the menu at the top of the window, select **File > Load**
- In the file open dialog, navigate to your **DirectOutput > config** folder and select the file GlobalConfig_B2SServer.xml. (If you can't find this file, create one as described below.)
- Click the Logging tab
- Check the box "Enable logging"
- Type **.\DirectOutput.log** into the Log File box
- On the menu at the type, select **File > Save**
- Select the same file we started with (GlobalConfig_B2SServer.xml) and confirm

DOF will now create a log file called **DirectOutput.log** in your **Visual Pinball > Tables** folder each time you run a game. To test this out:

- Start a new VP session
- Load a table that includes DOF effects
- Run the game (press F5)
- As soon as the game finishes loading, you can quit ("Q" then "Q") and close VP
- Open your **Visual Pinball > Tables** folder
- Look for a file called **DirectOutput.log**
- Bring up the file's properties and check its Date Modified: it should be moments ago, since it should have been created or updated during the VP session you just finished
- This is an ordinary text file, so you can open it in Notepad to view its contents

Look through the file to see if there are an ERROR or EXCEPTION messages. If so, read the messages to see if they mean anything to you. Many of these messages are of a technical nature that are meaningful only to someone familiar with DOF's program source code, but some will tell you about straightforward problems like missing files. If the messages give you any indication what's wrong, try correcting the indicated problem; if there are error messages that you don't understand, you can post them to the forums and see if anyone there can decipher them.

Manually create the DOF global config file

If you downloaded my DOF R3++ version listed in the setup instructions, it should have included a **config** folder inside your main DirectOutput folder, and a subfolder under that called **examples**. That should contain a file called **GlobalConfig_B2SServer.xml**. You can simply copy this file to the **config** folder.

If you don't see the **config > examples** folder or the **GlobalConfig_B2SServer.xml** file, you can create them manually:

- Go to your DirectOutput folder in Windows
- Right-click in the background area of the window and select **New > Folder**

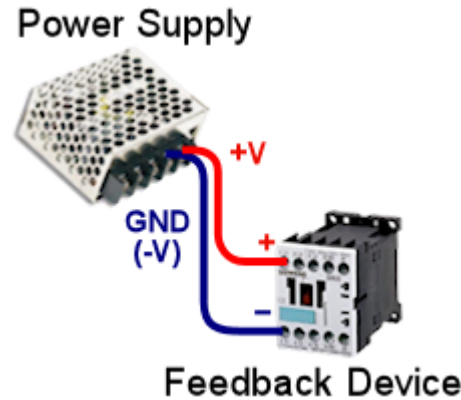
from the context menu

- Name the new folder **config**
- Open the **config** folder
- Download this file into the **config** folder: GlobalConfig_B2SServer.xml

47. Feedback Device Wiring

The basic principle of operation for all feedback device wiring is that the output controller acts as a switch for the device's power connection.

Before we get into that, though, let's start small, by just thinking about how you'd set up the power wiring for a device *on its own*, without having any sort of output controller or switching involved. This would be a simple matter of hooking up wires between the (+) and (-) terminals on the power supply and the device:



Wiring a feedback device directly to a power supply. A device wired this way will simply stay on all the time, and there will be no way for the software to control it.

Now, that's obviously useless for a pin cab, since it just makes the device turn on and stay on. The one exception is that you might wire some of the lamps in your front-panel buttons this way, if you don't care to place them under software control. And now that you mention it, this is exactly how the lights in my coin chute "reject" buttons are wired. There's really no need for them to ever turn off. Even on a real pinball machine, those are just hard-wired to power all the time. But for just about everything else, you'll want the software to control the device state, so this simple circuit plan simply won't do. But you already knew that - the whole point of this section is output controller wiring, after all!

So how exactly do we insert an output controller into this picture? The exact details can vary a little bit depending on the type of controller you're using - and we'll show you those exact details for each type of controller in the chapters ahead - but the principle is always the same. The basic idea is that we "cut" one of the power wires, and stick the output controller into that gap.

Basic plan for solid-state controllers

We're going to start with the "solid-state" controllers - LedWiz, PacLed, and Pinscape. We call these solid-state controllers because they use transistor switches to turn the attached devices on and off. We start with these because they all work roughly the same way, and they all have a fairly simple approach to connecting devices.

The alternative to a solid-state controller is a relay-based controller, such as a SainSmart board. And even for those, most of what we're going to cover for the solid-state controllers applies, which is another reason we're starting with those. We'll cover the specifics of the relay boards later on, but I hope you'll read through the solid state section first to get a mental picture of how all of this works in general terms. Just be aware as you read the solid-state material that there will be a few little differences for the relay boards. We'll provide the exact circuit diagrams for the relay boards in Chapter 51, SainSmart Relay Board Setup, so treat the circuit

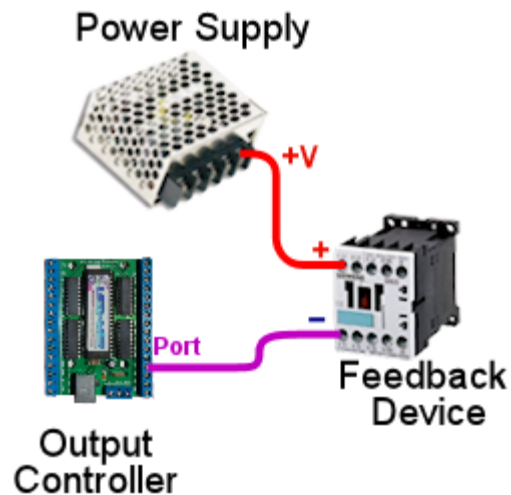
diagrams in the solid-state section as "theory" rather than practice.

The same goes if you're using a relay board to "boost" an LedWiz or PacLed's power. I sincerely hope you're not thinking of using a relay board to boost Pinscape's power, though, because the expansion boards are such a better way to do that.

If you're using a solid-state controller - Pinscape, LedWiz, or PacLed - you'll be able to follow the wiring diagrams in this section pretty literally, so you can pay as much attention to the details as you like. The same applies if you're using any of these boards with a solid-state booster board, such as one of Zeb's booster boards or a Pinscape DIY booster circuit.

Okay, let's get back to this idea that we're going to "cut" into one of the power wires to interpose the output controller as a "switch" that can turn the device on and off by controlling its power connection.

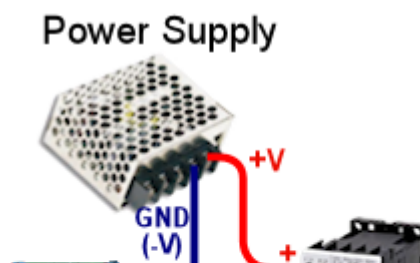
For all of the "solid-state" output controllers (Pinscape, LedWiz, PacLed), you don't have any choice about which wire has to be cut. The cut *always* goes in the (-) wire for these controllers:

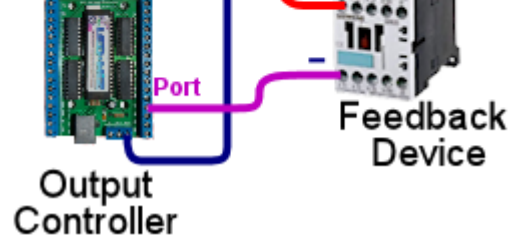


Basic feedback wiring plan for LedWiz, PacLed, and Pinscape. Connect the feedback device's (+) terminal directly to the power supply (+) terminal.

As you can see, we did as described above: we "cut" the wire between the feedback device's negative (-) terminal and the (-) terminal on the power supply, and we inserted the output controller into the gap. Specifically, we connected the device's (-) terminal to a "port" on the output controller.

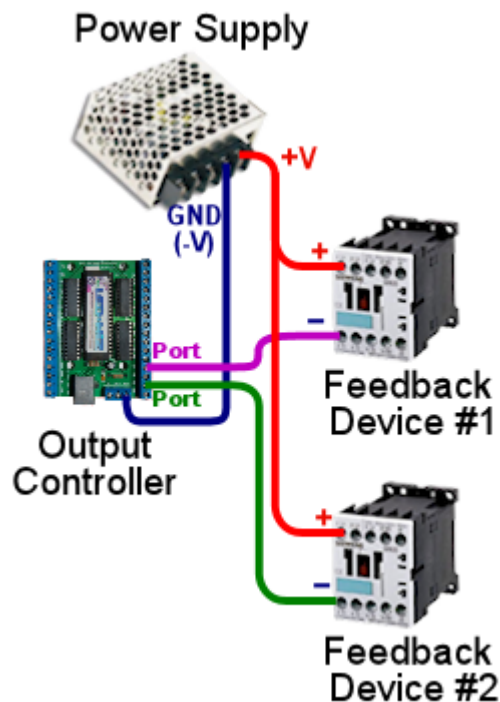
If you're paying close attention, you might want to point out a big error in this diagram: we lost a piece of wire here! The connection that we had before to the (-) terminal on the power supply is nowhere to be seen! You're right - that connection is definitely required. The reason we didn't show it is that it's actually part of the output controller's basic wiring that you set up when you install the output controller itself. It's not something you have to add for each device. For the sake of completeness, though, let's put it back into the picture:





A more complete view of the basic feedback wiring plan, showing the output controller's wire back to the (-) terminal on the power supply. This was elided in the earlier diagram because it's part of the basic wiring that you do when you set up the output controller, not something you have to add for each device you connect.

To clarify what we mean about not having to add this wire for every device, let's look at how you'd add a second device to this setup. To add a second device, you'd just add the two wires we showed in the first picture: the wire from the power supply (+) to the device (+), and the wire from the device (-) to the output controller port. For a second device, you'd of course use a different port, since each device must have its own separate port.



Adding a second device to the controller. The new device is wired just like the first device, using a separate port on the controller. Note that we don't have to add another wire back to the power supply (-) port: all devices share a single "ground" connection.

There are two things to notice here. The first is that, as we pointed out, you don't have to add another connection to the power supply's (-) terminal. The single "ground" connection that you set up when you install the controller is shared by all devices, so adding a new device only requires the two wires: power supply (+) to device (+), and device (-) to output controller port. The second thing to notice is that we "daisy-chained" the power supply connections. We didn't have to run a separate wire all the way from the second device back to the power supply; it's sufficient to connect them through the same wire. This can save you a lot of wire, since you'll probably have a few devices that are fairly far away from the power supply.

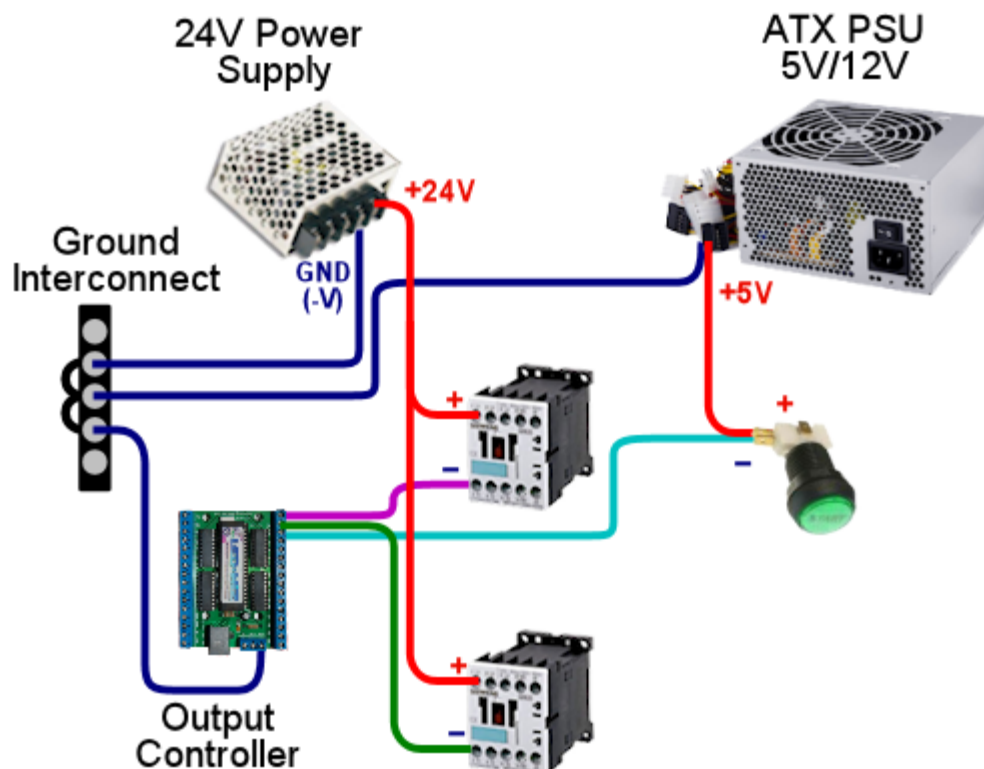
This is all well and good if you want to hook up a bunch of devices that run on the same voltage. But what about devices that run on different power supplies that use other voltages? If you already read through Chapter 45, Power Supplies for Feedback, you know that you might need as many as five or six different voltages if you're using a diverse mix of devices:

- 5V for flasher LEDs
- 6.3V for incandescent button lamps
- 12V for the shaker motor and gear motor
- 14V for #161 lamps in the big square buttons
- 24V for contactors and chimes
- 30V or more for the the replay knocker

So how do you hook up all of these different devices, when each one has to be connected to its own power supply? What happens to that common wire back to the (-) on the power supply when we have five power supplies?

Happily, this turns out to be easy. If you go back and review the section on Chapter 45, Interconnecting Grounds in the Power Supplies chapter, you'll find the answer. You're going to explicitly wire all of your power supply (-) terminals together when you set up the power supplies. So that wire that connects from the controller to the (-) terminal on the one power supply actually connects to the (-) terminal on *all* of the power supplies. This important power supply setup step is explained in detail in Chapter 45, Power Supplies for Feedback.

Now that we've settled the question of the (-) wire, it becomes straightforward to connect a mix of devices that use different power supplies. You just repeat that same formula for each device and its particular power supply: device (+) to power supply (+), device (-) to output controller port.



*Adding a third device that uses a different voltage and a different power supply.
The new device is wired just like the other devices, with its (+) connected to its*

power supply (+) and its (-) connected to an available output port. All power supply (-) terminals are connected together at a common ground interconnect point, so the output controller's GND connection only has to connect to the common interconnect point.

Hopefully, it's clear now how this generalizes if you wanted to add a third or fourth power supply to the mix. I don't want to elaborate the diagram any further to show even more devices and power supplies, since it's already getting pretty busy (and I'm running out of colors for the wires), but you just apply the same principle for each additional power supply:

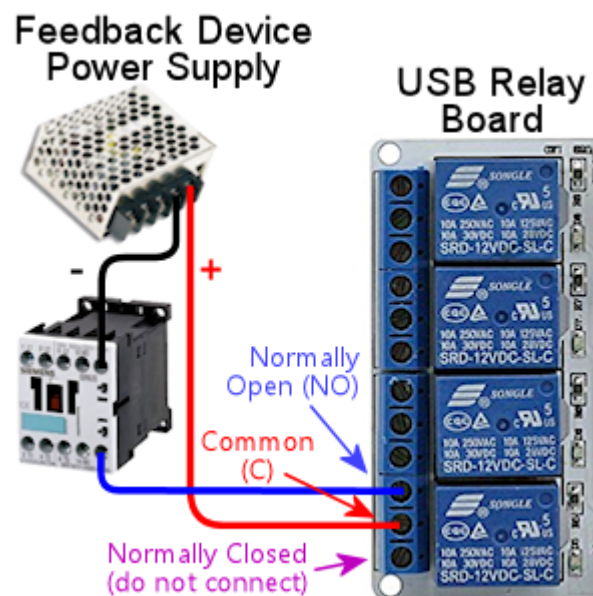
- Connect the power supply (-) to the common ground interconnect point
- For each device using the power supply's voltage, connect the device's (+) terminal to the power supply (+) terminal, and connect the device's (-) terminal to a free output port

Basic wiring plan for relay-based devices (SainSmart)

If you're using a SainSmart USB relay board, or some other type of relays, the wiring plan is *almost* the same as shown above. There are two slight differences, though.

First, with a relay, you can "cut" *either* the (+) or (-) supply wire to insert the relay into the circuit, but it's considered better engineering practice to insert the relay into the "+" wire. That's the way it's shown in the diagrams in the Chapter 51, SainSmart Relay Board Setup, which is part of why I warned you earlier that the relay-based wiring diagrams would look a little different when you got to them. (The reason it's better to cut the (+) wire is that this leaves the device without any live positive voltage connected when it's switched off. That's safer in case you accidentally touch one of the wires, or there's an accidental short circuit.)

Second, relay boards *don't* have that single common ground wire that runs back to the shared (-) terminal interconnects. Instead, you simply insert the relay into the circuit as though it were an ordinary switch (which, in fact, it is!).



See Chapter 51, SainSmart Relay Board Setup for more details on the SainSmart boards in particular.

Device-specific wiring considerations

You can use the basic wiring plan above with all types of feedback devices. However, some devices have additional, special wiring needs of their own that you have to add onto the basic generic plan.

The sections on the individual device types contain details on any special wiring requirements, but here's a quick overview of the things you have to watch out for.

LED current-limiting resistors

LEDs always require something to limit the amount of current (amperage) going through them. Without some kind of current limiter, an LED acts like a short circuit, which will make it overload itself or its power supply. The simplest and most common type of current limiter is a resistor placed in series with the LED.

For a detailed explanation of LED resistors, including a calculator to pick resistor sizes, please see Chapter 52, LED Resistors.

Diodes for coils, motors, solenoids

Any feedback device that uses a magnetic "coil" requires that you install a protective diode on the device. All coils inherently cause voltage spikes that can damage other components in your cab and can cause weird computer glitches. Diodes are used to suppress these spikes. They're cheap and install to install, so don't omit them.

The following all require diodes:

- Motors (shakers, gear motors, fans)
- Solenoids
- Contactors
- Replay knockers
- Chime coils
- Other pinball coils
- Relays

For details on what types of diodes to use and how to install them, see Chapter 53, Coil Diodes.

AC devices

It's rare for anyone to use an AC-powered device in a pin cab, but I wanted to mention this just in case: **Do not connect an AC-powered device to any of the solid-state output controllers.** All of these (LedWiz, PacLed, Pinscape) are designed to work with DC voltages only.

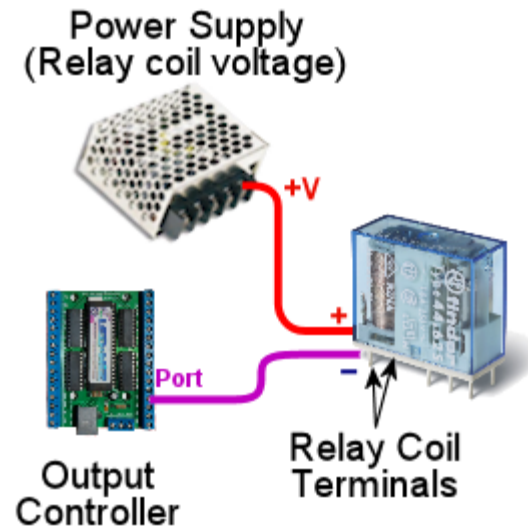
I recommend avoiding this issue entirely by sticking with the common DC-powered devices, but if there's something AC-powered that you really want to include in your cab, it's possible with a little extra work. What you have to do is connect an AC-compatible switching device *between* the output controller and the AC device. You have a few options for the switching device:

- Relay
- Solid-state relay
- H-bridge

If you're using a SainSmart USB relay board as your output controller, you already

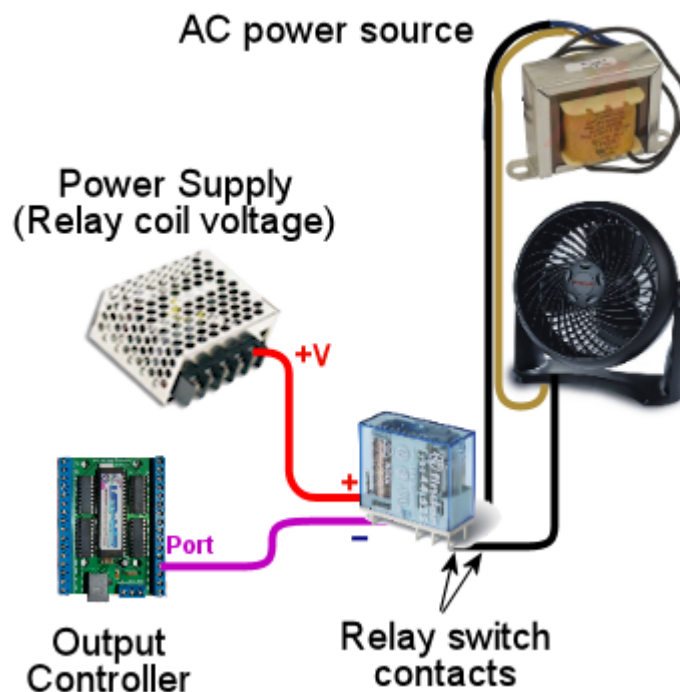
have this covered, because those use AC-compatible relays. You can connect an AC-powered device to the SainSmart the same way you'd connect any DC-powered device.

If you're using one of the solid-state controllers, I'd recommend keeping it simple by using a relay. Here's how. First, connect the relay coil to a port on your output controller exactly as though the relay coil were an output device in its own right:



Now connect the AC device to the "Normally Open" (NO) and "Common" (C or CMN) contacts on the relay. Run the following wires:

- From one AC power supply terminal to the relay NO terminal
- From the relay CMN contact to one power terminal on the device
- From the other power supply terminal to the other device power terminal



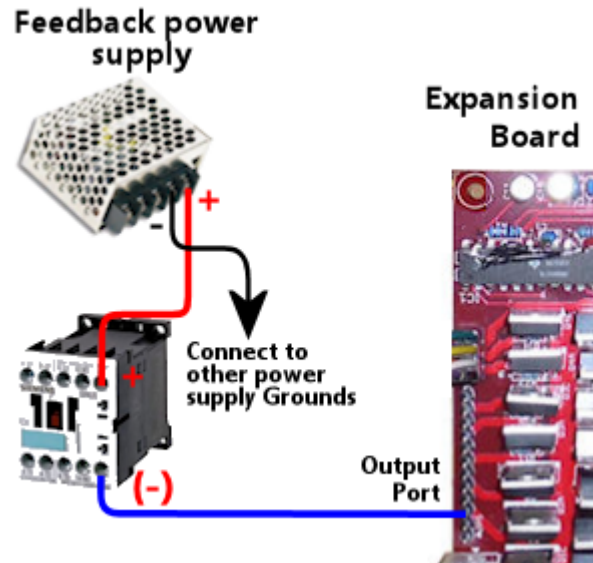
If you're familiar with solid-state relays or H-bridges, you can use one of those devices in place of a relay. Those generally aren't designed to be directly compatible with LedWiz or Pinscape-style switching, though, so you'll likely have to rig some additional circuitry to make that work. I'm going to leave that as an exercise to the

reader for those are so motivated, and recommend that everyone else go with the simple relay setup described above.

48. Pinscape Outputs Setup (Expansion Boards)

If you've built a set of expansion boards, you're past the hard part. Connecting output devices is fairly easy. The Pinscape boards use the same basic wiring plan as other solid state controllers, as described in Chapter 47, Feedback Device Wiring:

- Connect a wire from the **positive** terminal of the power supply to the feedback device's positive terminal
- Connect the feedback device's negative terminal to an output port pin on the Pinscape expansion board
- As described in Chapter 45, Power Supplies for Feedback, you should already have connected the power supply's negative terminal to the common ground interconnect for all of the feedback power supplies



Repeat this pattern for each device.

For devices that run on the same voltage, you can either run a wire from each device's (+) terminal all the way back to the power supply, or you can daisy-chain the positive terminals together. You can do whatever's most convenient for each device, and you can freely mix and match.

It's fine to mix devices that run on different voltages, and it's fine to connect them side by side on the same expansion board. Just connect each device to its appropriate (+) voltage supply.

Config tool settings

Make sure you've configured the Pinscape firmware for your expansion boards:

- Launch the Pinscape Config Tool
- Click the Settings icon for your device
- Near the top of the page, make sure that the System Type is set to Pinscape Expansion Boards

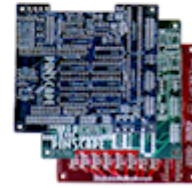
Settings

System type. Are you using this KL25Z on its own, or with a set of expansion boards?

[Help](#)



☐ Stand-alone KL25Z
(or your own custom boards)



☒ Pinscape Expansion Boards

- Make sure that the number of power boards and chime boards is set to match your physical configuration

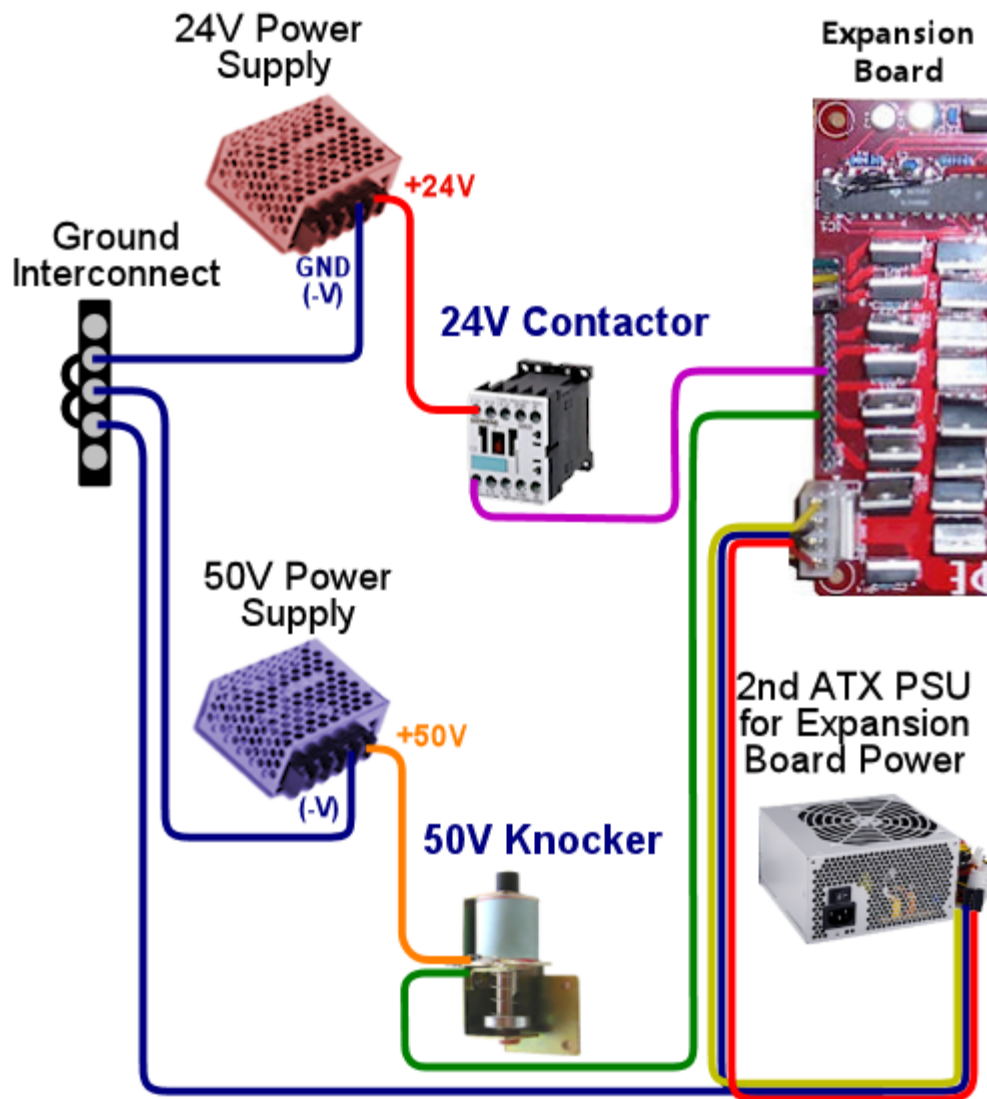
Mixing power supply voltages

The Pinscape boards switch power on the "Ground" (negative or 0V) side of the connection. This means that it's okay to mix different devices with different power supply voltages.

This is confusing to a lot of people because they only see power inputs on the Pinscape boards for 5V and 12V. But what if you want to connect a 24V device? Or a 50V device?

Here's how you have to think about this. The 5V and 12V power supply connections to the Pinscape boards are there *to power the Pinscape boards themselves*. Those connections *aren't* there to power the feedback devices. The feedback devices take their power *directly from the power supplies*, **not** from the Pinscape boards.

That means you don't have to connect the "+" from a 24V supply *anywhere* to the Pinscape boards, but you can still connect it to feedback devices that need 24V.



Use protective diodes if necessary

Most mechanical devices **require** protective diodes. These are required for anything with a coil: solenoids, contactors, replay knockers, motors.

See Chapter 53, Coil Diodes for details on what type of diodes to use and how to install them.

Diodes are critical! Coils and motors can damage your PC motherboard and other components if diodes aren't used.

Fuses

Many cab builders include a fuse in each device output circuit. This isn't strictly necessary, but I think it's a good idea for the higher-powered mechanical devices. Fuses can help protect the Pinscape board from overloads caused by wiring faults and device malfunctions, reducing the chance that the board will be damaged if something like that goes wrong. See Chapter 84, Fuses for details.

Where to connect what

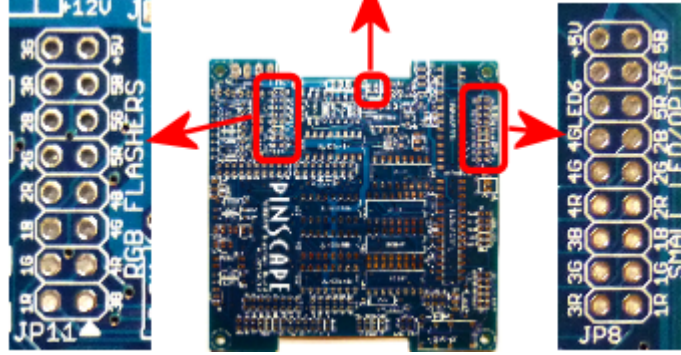
The expansion boards have a number of output ports that are designed for particular device types. It's best to attach devices to their special ports when present, since that takes the best advantage of the special design features on the boards.

Here's a quick summary of where to attach each common device type. If you're not familiar with any of these, we'll look at each type in depth over the next few chapters.

Device	Where to connect
Beacon (rotating police light)	Power board
Bell (one-shot type)	Chime board
Bell (reciprocating type)	Power board
Bumper contactors/solenoids	Power board
Button lamps (front panel)	Power board
Chimes	Chime board
Fan (backbox topper)	Power board
Flasher LEDs (two sets can be wired in parallel)	Main board, JP11 (RGB Flashers)
Flipper button LEDs	Main board, JP8 (Small LEDs)
Flipper contactors/solenoids	Power board
Gear motor	Power board
Pinball bumper assemblies	Chime board
Pinball slingshot assemblies	Chime board
Replay knocker	Main board, JP9 (Knocker pin)
Shaker motor	Power board
Slingshot contactors/solenoids	Power board
Strobes	Main board, JP9 (Strobe pin)
Undercab LED strips (up to about 3 meters without separate amplifiers)	Power board
Addressable light strips	N/A; needs separate controller

Main board output ports

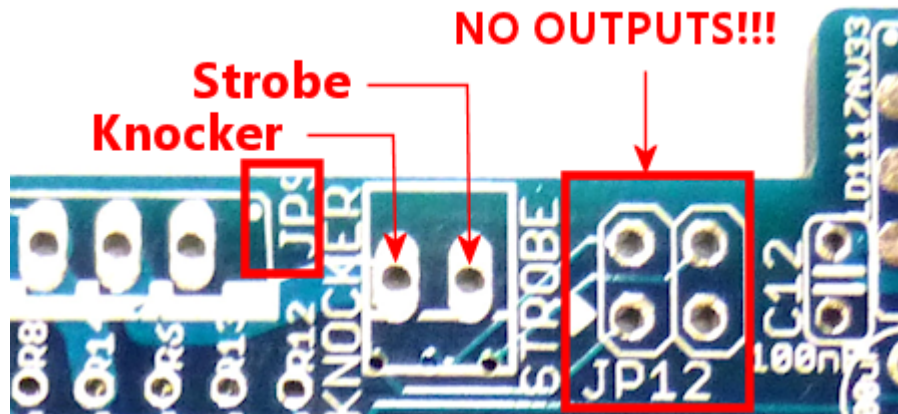




Output ports on the main expansion board. JP11 (left) is for five RGB flashers; JP9 (top) is for the replay knocker and strobe; and JP8 (right) is for small (20mA) LEDs.



Warning! Don't connect a strobe or any other output device to any of the pins on JP12. JP12 is **not** an output port header. The printing on the board is so packed together in that area that it's easy to read the STROBE label as part of JP12. It's not! The STROBE pin is on JP9, right next to the KNOCKER pin.



Be careful about where you connect the strobe and knocker. Both of these connect to the pins on JP9. Don't connect any output device to JP12. The pins on JP12 connect directly to GPIO pins on the KL25Z, so connecting them to power is extremely likely to damage or destroy the KL25Z.

Main board JP11 (RGB Flashers) is designed for high-power RGB LEDs, for flasher lights. See Chapter 56, Flashers and Strobes for information on the type of LED most people use. This header has 15 outputs, which is enough to connect the standard complement of 5 flasher LEDs (as each RGB LED takes up three channels, one each for red, green, and blue).

The pins on the header are labeled 1R, 1B, 1G, 2R, 2B, 2G, etc. "1R" stands for the Red channel on the first LED, for example. This is purely a suggestion for how to connect the color channels; you can rearrange the connections if necessary. If you connect them according to the labels, though, it will make the software setup easier, since this ordering matches the default settings in the firmware.

This connector also provides a +5V power supply pin. The idea is that you can connect a single 16-wire ribbon cable between this header and a matching header on your flasher panel to provide all required wiring for the flashers. This is just here as a convenience; if for some reason you'd prefer to provide your own power supply connection to your flasher panel, simply don't connect anything to the +5V pin on this header.

You don't strictly have to connect flashers to these outputs, although if you're going to use flashers in your system at all, this is the best place to connect them. If you're *not* using flashers, though, you can re-purpose these outputs for other types of devices, **as long as you stay within the limits of 50V and 1.5A** per output. Don't connect anything that draws more than 1.5A, as that can potentially destroy the IC chips that drive these outputs. For anything that takes more than 1.5A, always use the Power Board outputs instead.

Main board JP9 (Knocker/Strobe) has one pin dedicated to the replay knocker, and one pin dedicated to a "strobe" output. These assignments are printed on the board next to the pin header to help you identify which is which. See Chapter 59, Flippers, Bumpers, and Slingshots and Chapter 56, Flashers and Strobes for notes on these device types.

The strobe output is driven by exactly the same chip that controls the flasher outputs, so it has the same power limits (50V and 1.5A). You can use it for other purposes besides a strobe if you prefer, as long as you stay within those limits.

The knocker output is driven by a MOSFET that can handle very large loads, which is necessary because real replay knockers are designed for 50V supplies and draw about 3A. However, it's not great as a general-purpose output, because it has a "time limit" circuit that prevents the output from staying on for more than about 2 seconds at a time. The time limit is specially designed to protect replay knocker coils against common software faults in Visual Pinball that can leave devices stuck on; this problem has destroyed replay knockers on more than a few virtual cabs, but you're relatively immune to this with the Pinscape boards because of the built-in time limiter circuit. The downside is that it means that the port is really only suited for a replay knocker, since you wouldn't want similar time limits on most other types of devices. The one bit of re-purposing that could be appropriate for this output is that you could use it for a music chime, since chime coils have the same vulnerability to overheating that replay coils have.

See the warning above about the proper location of the STROBE output.

Main board JP8 (Small LED/Opto) has 16 pins designed for low-power LEDs. These are intended primarily for controlling LEDs in the flipper and MagnaSave buttons; see Chapter 55, Button Lamps for how to set those up.

As with the RGB flasher header (JP11), these pins are labeled with color channels - 1R, 1G, 1B, 2R, 2G, 2B, etc. And as with JP11, you're free to ignore the labels if you prefer, but following them will save you a little software setup time.

In addition to labeled outputs for 5 RGB LEDs, there's an extra pin labeled LED6. This is a spare that you can use for an additional monochrome LED, or just leave unconnected. For that matter, there's no common use for a 5th RGB LED, since a typical cab only has the four flipper/MagnaSave buttons. You can repurpose the 5R-5G-5B outputs for whatever other small LED uses you can think of, and you're free to treat them as separate channels and connect each one to a monochrome LED. The RGB-ness is only a suggestion.

This header provides a +5V power supply pin that you can use to power the attached LEDs.

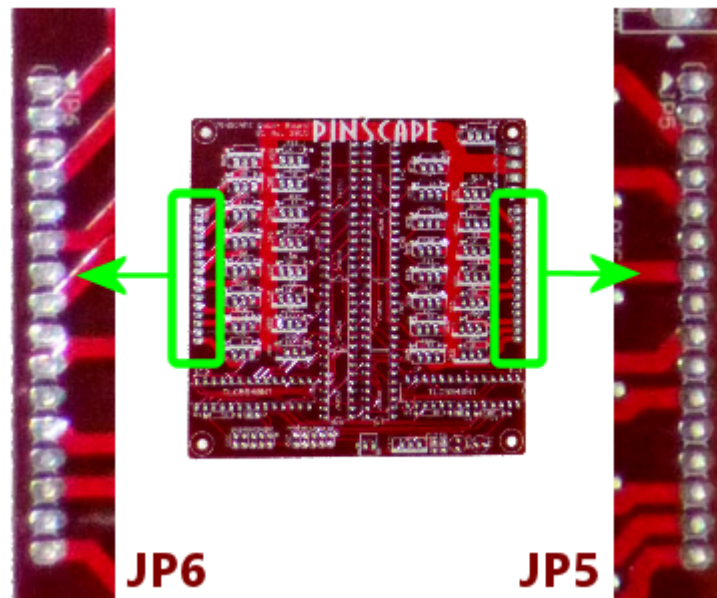
This header has a special feature for LEDs: it has a built-in current limiter for each channel, which eliminates the need to use a separate external resistor for attached LEDs. (See Chapter 52, LED Resistors.) You can safely connect LEDs directly to these ports, with no resistors required. The current limit is determined by the resistance (Ohms) value of resistor "R5" on the main board. If you used the default 2.2K resistor from the parts list, the current limit is set at 20mA, which is perfect for most

small LEDs.

The term "Opto" in the label, by the way, refers to optocouplers. Optocouplers have internal LEDs with the same general properties as the small LEDs that these outputs are designed for, so you could connect an optocoupler to any of these ports in lieu of an LED. That would allow you to control another circuit through the opto. I don't have any particular applications for this in mind - it's just an option for people who like tinkering with electronics, since it gives you a way to control just about any custom circuit via a Pinscape output.

Power Board output ports

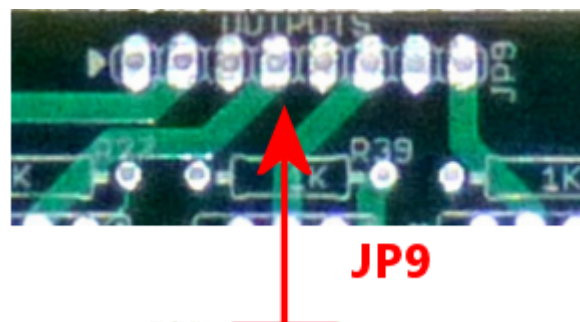
All 32 of the Power Board outputs are general-purpose, high-power outputs that you can connect to almost anything. These outputs are suitable for lights, motors, solenoids, and just about anything else commonly used in a pin cab. They can handle devices that use power supplies up to 60V and current up to 4A.



Power board outputs. These are general-purpose outputs suitable for almost anything in a pin cab. Each output can handle 4A at 60V.

Chime board output ports

All 8 of the Chime Board outputs are high-power outputs with time-limiter circuits of the same type used in the replay knocker output on the main board. As with the replay knocker, these are designed to protect pinball coils from software faults on the PC that could leave an output stuck on. They're nominally for "chime units" (see Chapter 64, Chimes and Bells), but they're really appropriate for any sort of standard pinball coil, such as bumpers and slingshots. You wouldn't want to use them with flippers, however, as you certainly do want to allow flippers to be held on for long periods.





Chime board outputs. These outputs have the same type of time limiters as in the replay knocker output on the main board, to protect attached pinball coils from overheating if the software on the PC crashes and leaves outputs stuck on. Outputs are automatically turned off if stuck on for more than about 2 seconds. Each output can handle 4A at 60V.

Power limits

Here's a summary of the power limits for the various outputs on the expansion boards. The amperage limits are per port.

The Timer column indicates whether or not the output has a timer circuit that cuts off power after it's been on continuously for more than a couple of seconds. These apply to the replay knocker output on the main board and all outputs on the chime board. The timer circuit is there to prevent software crashes from leaving a pinball coil stuck "on", which can overheat and destroy those coils. The timed outputs are good for pinball coils, where you only ever want momentary activation; they're not good for anything that you actually want to leave running for long periods, like shaker motors, gear motors, fans, beacons, lights, etc.

Board	ID	Description	Max Limits		Timer
Main	JP8	Small LEDs	18V	20-50mA	No
Main	JP9	Knocker	60V	4A	Yes
Main	JP9	Strobe	50V	1.5A	No
Main	JP11	RGB Flashers	50V	1.5A	No
Power	JP5, JP6	General Purpose	60V	4A	No
Chime	JP9	Chimes	60V	4A	Yes

DOF Setup

To set up your Pinscape unit with DOF, you of course have to install DOF on your PC first. See Chapter 46, DOF Setup for instructions.

Once the DOF software is installed, you use the DOF Config Tool to tell DOF that you have a Pinscape unit, and to tell it which output ports are wired to which feedback devices.

- Open the DOF Config Tool in your browser
- Click the My Account tab

- Set **Number of Pinscape devices** to 1 (or if you have more than one KL25Z running the Pinscape, select the appropriate number of devices instead). The number of expansion boards connected to the same KL25Z doesn't matter here - if you have a main board and two power boards and two chime boards, you should still enter **1** for **Number of Pinscape devices**, since they're all part of the same USB connection from the PC's perspective.
- Set **Number of KL25Z devices** to **0** (see below if this seems confusing)
- Save changes
- Go to the Port Assignments page
- Select "Pinscape 1" in the Device drop list
- Go through the port list, assigning each port number in the DOF list to the device that you wired to the corresponding expansion board port

The port list in the Port Assignments page uses the same port numbering (Port 1, Port 2, etc) that's shown in the Pinscape Config Tool output port list. This **isn't** a physical pin number from any of the headers. It's the abstract port number from the output port assignment list.

To figure out what DOF's "Port 1" or "Port 2" means in terms of the physical expansion board pins, you have to look at the output port list in the Pinscape Config Tool's Settings page. In the output list, find the same port number shown in DOF - if you're looking for DOF's "Port 1", you want the first row, #1, in the Pinscape output list. Trace across the row to find which physical pin that port is assigned to. If you want to see a picture of where that pin is physically located, click the pin name in the row - that will pop up the pin selector, which will show the pin location highlighted on a picture of the appropriate expansion board.

Is it a "Pinscape device" or "KL25Z device" or both?

The online DOF Config Tool has a confusing bit of terminology in the device setup section on the "My Account" page. In the list of devices, you'll find separate entries for "Number of Pinscape devices" and "Number of KL25Z devices".

For Pinscape boards, use **only** the **Pinscape devices** option. Always leave "Number of KL25Z devices" set to zero (0). This applies whether you're using the expansion boards or a standalone KL25Z.

This is confusing because the Pinscape software does happen to run physically on a KL25Z board, so it might seem like you should enter the same number for both line items. Don't. Pretend that you've never heard of a KL25Z and that you have zero of them.

The Config Tool has the "KL25Z devices" line item for historical reasons that date back to the first version of the Pinscape software, when it was limited to 32 output ports. It should more properly be titled "Number of Pinscape v1 devices", because that's what it really means. In any case, just ignore it and leave it set to zero.

49. Pinscape Outputs Setup (Standalone KL25Z)

The standalone KL25Z (without the expansion boards) is capable of controlling feedback devices such as LEDs, motors, and solenoids. However, there are some important limitations you should be aware of:

- The GPIO pins on the KL25Z have *extremely* low power limits, so you can't ever connect a feedback device directly to a GPIO pin. **All** devices have to be connected through "booster" (amplifier) circuits, which you have to build separately and connect to the KL25Z. You really do need boosters for **everything**, even small LEDs, because the GPIO power handling limits are simply too low to connect anything directly.

There are several options for booster circuits that you can use for different types of devices. Full instructions are provided later in this section.

- The number of devices you can connect is limited by the number of available GPIO pins, because each output device requires its own dedicated GPIO pin. The KL25Z has about 50 GPIO pins in all, but remember that many of them will be needed for other purposes, such as button inputs and plunger sensor connections. In the default configuration, there are only about 20 pins available for feedback devices. You can change the pin assignments with the Config Tool, so you can take some pins that are assigned by default to other functions and reassign them as feedback outputs, but that will of course reduce the number of pins available for those other functions.
- The KL25Z only has 10 PWM channels. PWM is needed if you want to control the brightness of an LED or the speed of a motor; without PWM, a GPIO pin can only act as a simple on/off switch. 10 channels isn't enough for the standard five-flasher setup, since each RGB flasher needs three PWM channels (one each for red, green, and blue).

If you can live within these limits, the standalone KL25Z makes a great output controller at low cost. If you need more outputs in general or more PWM outputs in particular, you might want to consider adding the Pinscape expansion boards - which were specifically designed to overcome all of these limits - or one of the commercial I/O controller devices. See Chapter 86, Expansion Boards Overview for more on the expansion boards, and Chapter 12, I/O Controllers for a survey of the commercial alternatives.

An additional option for adding more outputs, if you run out of available GPIO pins, is to simply add a second KL25Z. The Pinscape software is set up so that you can run multiple KL25Z's in the same system.

Viewing and assigning output ports

Most of the GPIO pins on the KL25Z can be assigned to different purposes using the Pinscape Config Tool. The default configuration for a standalone KL25Z (without the expansion boards) assigns 22 pins as output devices, but you can add more output ports by reassigning pins assigned to other purposes, such as button inputs.

To view the list of ports currently assigned as output pins:

- Run the Pinscape Config Tool
- Go to the Settings page
- Scroll down to the **Feedback device outputs** section

That will show you a list of all of the pins assigned as output ports. Each port has a port number, which is used to identify the port in the DOF Config Tool. That port number is just an abstract label, for DOF's sake, that doesn't mean anything in terms of KL25Z hardware. To find the KL25Z hardware pin, scan across the row to find the GPIO pin.

Each row also show a port type, which can be Digital Out or PWM Out. Digital ports are simple on/off ports, with no ability to control brightness or intensity. PWM ports can set different brightness levels on the output, so they're best for LEDs and other lighting devices. ("PWM" stands for "pulse width modulation", which means that the port switches on and off very rapidly. The ratio of "on time" to "off time" determines the effective brightness or intensity.) PWM ports are also good for shaker motors and fans, since the intensity control can be used to adjust the speed of the motor.

The KL25Z has an inherent hardware limit of 10 PWM ports. Further, only certain pins (about 20 of them) have PWM capability at all. There's no need to memorize which pins are which; the Config Tool's pin assignment dialog always shows you the available pins of each type.

A "Virtual" port is simply a port that's not connected to anything. It still shows up over the USB connection, so DOF and other PC software can access it as though it were actually there. But when DOF sends it a command, it'll just harmlessly ignore the command. Virtual ports let you satisfy DOF's expectations that a certain number of ports are present even if you don't have enough physical GPIO pins available for all of the ports DOF expects.

To add a new output port, go to the feedback device outputs section as described above, then:

- Find one of the ports that's currently marked "Virtual". If there aren't any of those, click the "+" icon in the blank row at the very end of the list. That will add a new Virtual port.
- Click the pin assignment box (currently "Virtual") in the port row you want to assign. This will open a pin selector dialog.
- In the pin selector, click the type of port you want to add at the right (Digital Out or PWM Out).
- Select the GPIO pin you want to assign.

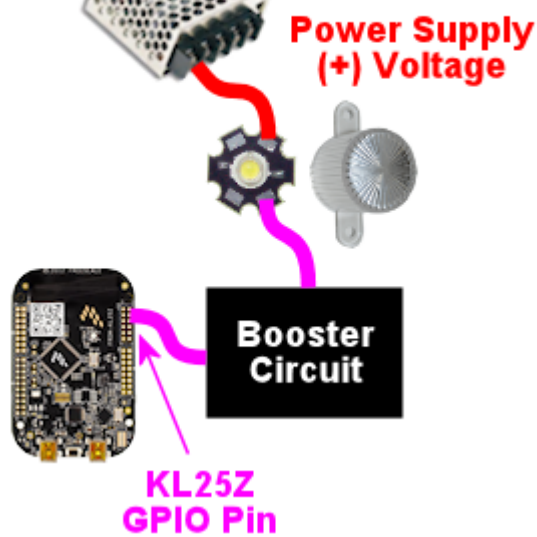
Important! A pin can only be assigned to one function at a time. The dialog will allow you to select pins that are already assigned to other functions, but if you do, the pin will show a little error icon (❗) next to the pin. You **must** resolve any such warnings before the device will work properly.

To determine what a pin warning means, simply click the warning icon. This will pop up a dialog explaining what's wrong. The usual problem is that you've assigned the same pin to two or more functions. That's simply not workable at a hardware level, so you'll have to remove the conflicting assignment(s). The warning message in the popup will tell you exactly where the conflicting use is assigned. Go to the appropriate section of the Settings page and remove that conflicting assignment.

Wiring KL25Z outputs

The basic wiring plan for a KL25Z output looks like this:





The complication is that little box labeled "booster circuit". That's required because the KL25Z itself only allows a tiny, tiny amount of current to flow through each GPIO pin. Too much current will damage the KL25Z by overheating its delicate internal wiring.

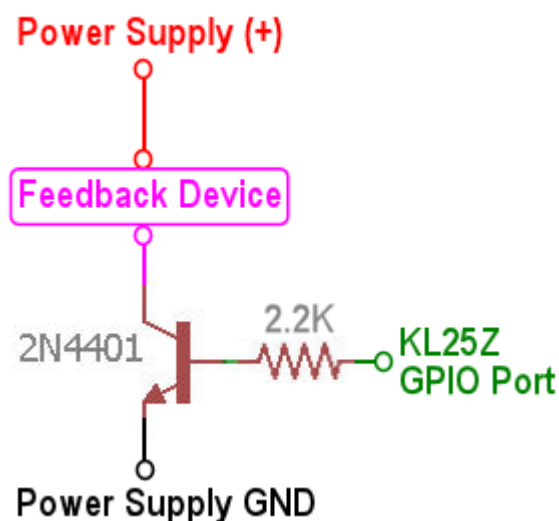
The reason we show the booster circuit as a little box rather than an electronic schematic is that you have multiple options for what to put there. We'll look at some common options in a moment.

Note that the diagram above also leaves out other details specific to the type of output device. For an LED, you'd need a current-limiting resistor somewhere in the wiring; see Chapter 52, LED Resistors. For a coil, solenoid, motor, etc., you'd need a diode to protect against the magnetic field effects; see Chapter 53, Coil Diodes. See the descriptions of the individual types of feedback devices for more details on each one's special needs.

Remember, the exact KL25Z pins that correspond to output ports are under your control! That's why we're not just printing a list here - there is no fixed list to print. To find out what's currently assigned in your setup, look at the Settings page in the Pinscape Config Tool as described above.

NPN booster

This is a simple and inexpensive booster you can build for a low-power device, up to about 600mA at 40V. That's enough for a small incandescent lamp (e.g., a #555 bulb in a front-panel button), a flasher LED, or a small relay.



The power supply (+) and GND connections are meant to connect to the secondary power supply that you're using for the feedback devices. Note that you should provide a wire directly to the power supply ground from the transistor's emitter, as shown, rather than connecting it to a Ground pin on the KL25Z. The KL25Z ground pins ultimately do all connect back to the power supply grounds, but they go through the computer's USB port, which has limited current carrying capacity. It's important to bypass that with a wire directly to the power supply ground to avoid forcing too much power through the USB port.

The 2N4401 is a common transistor that should be easy to find, but there's nothing magical about that particular part. There are hundreds of similar "small signal transistors" that can be substituted. Common substitutes: BC547, 2N3904, PN2222. If you do substitute a different transistor, check its data sheet for the power limits, since the limits quoted above (600mA, 40V) are specifically for the 2N4401, and other transistors might have different limits. The maximum values to look for in the data sheet are the "Absolute Maximum" ratings for:

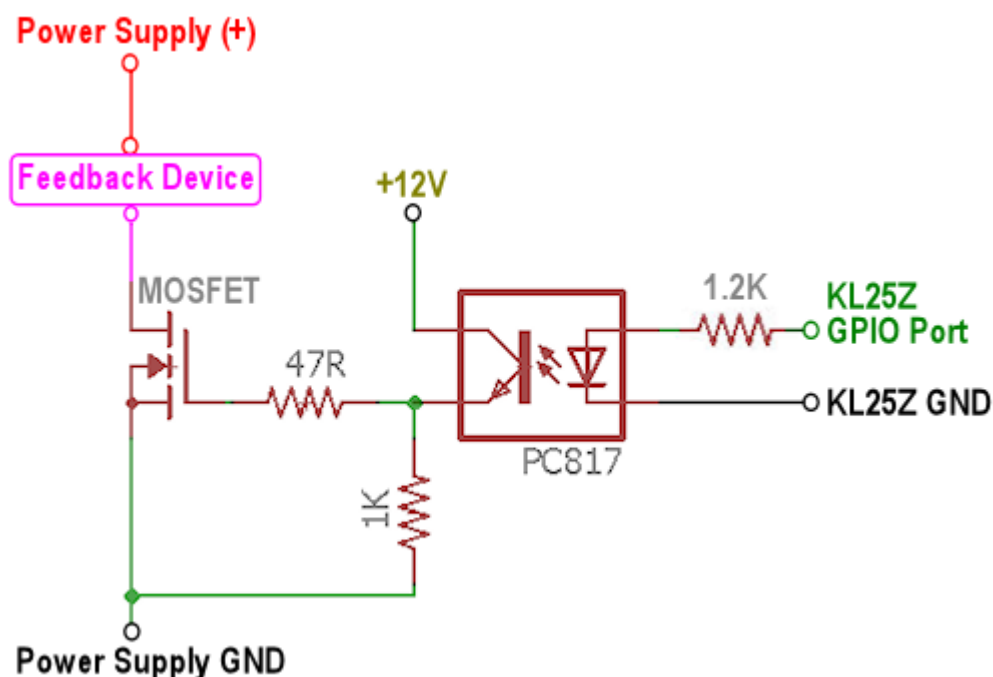
- Collector-Emitter Voltage (V_{CE0}): maximum power supply voltage for the feedback device
- Collector current, continuous (I_C): maximum feedback device current

Any ordinary 2.2K resistor will work. (The smallest resistors are usually 1/8 Watt, which is perfectly fine for this job. Higher wattage resistors will work as well but are unnecessary.)

DIY MOSFET booster

This is my go-to booster circuit. It's the one used in the Pinscape power boards. It's versatile enough for just about anything you'd want to put in a pin cab, including high-power toys like shaker motors and replay knockers. The exact power limit depends on the specific MOSFET you choose, but common MOSFETs are readily available that can handle way more than you'd ever need, upwards of 10 or 20 Amps.

This circuit is more complex than the NPN booster above, but it's still pretty simple - only five parts.



If you need help decoding the schematic, see A Crash Course in Electronics.

The resistor slots can all be filled with ordinary resistors with the specified "Ohms" values. There's no special Wattage rating needed for any of these.

Which MOSFET to use? Here's a list of parts I've tried that work well:

- BUK7575-55A
- FQP13N06L
- FQP30N06L

But lots of other MOSFETs will work just as well. Any N-channel type sold by an Arduino or robotics company will probably be suitable, since robotics projects often use these parts exactly the same way we do (and for the same reasons). If you want to cast a wider net by looking on Mouser, the basic type of part you need is an N-channel enhancement-mode MOSFET - but that turns up about 8,000 matches on Mouser, so here are some more specific characteristics to look for:

- Low "on" resistance ($R_{DS(on)}$), below 1Ω (preferably something like $100m\Omega$)
- Drain-source voltage (V_{DSS}) sufficient for your feedback device power supply, preferably above 40V
- Continuous drain current (I_D) sufficient for your feedback device's needs, preferably above 10A
- Through-hole package (for easier soldering)

Here's a Mouser search for those characteristics. This still matched about 1,400 parts when I tried it, so it doesn't exactly narrow things down to a trivial selection, but I'd sort by price, pick one of the cheaper ones, and scan the data sheet to make sure it looks like a suitable part for logic applications.

[www.mouser.com/Semiconductors/Discrete-Semiconductors/Transistors/MOSFET/_/N-ax1sfZ1yzvvqx?](http://www.mouser.com/Semiconductors/Discrete-Semiconductors/Transistors/MOSFET/_/N-ax1sfZ1yzvvqx?P=1z0y3zrZ1yiaumvZ1z0z63xZ1z0y4ci&Rl=ax1sfZgjdhp3Z1yw78ezZ1ypyijjSGgjdhozZ1yw76gfZ1yvi0)

[P=1z0y3zrZ1yiaumvZ1z0z63xZ1z0y4ci&Rl=ax1sfZgjdhp3Z1yw78ezZ1ypyijjSGgjdhozZ1yw76gfZ1yvi0](http://www.mouser.com/Semiconductors/Discrete-Semiconductors/Transistors/MOSFET/_/N-ax1sfZ1yzvvqx?P=1z0y3zrZ1yiaumvZ1z0z63xZ1z0y4ci&Rl=ax1sfZgjdhp3Z1yw78ezZ1ypyijjSGgjdhozZ1yw76gfZ1yvi0)

As with the NPN booster above, the power supply (+) and GND connections are meant to connect to the secondary power supply that you're using for the feedback devices. (Don't connect "Power Supply GND" through a KL25Z Ground pin.)

If you're connecting a device with any sort of coil (including solenoids, relays, contactors, and motors), be sure to use a diode with the device, as explained in Chapter 53, Coil Diodes.

Note: the old version 1 Pinscape Build Guide had a similar circuit plan for a MOSFET booster, but that plan added a Darlington transistor chip between the GPIO pin and the PC817. This newer version dispenses with that extra chip. The purpose of the Darlington was to boost the weak signal from the KL25Z to better drive the opto, because I was concerned in the early days about the optos being sensitive enough for the KL25Z to drive them directly. Experience has shown that the extra boost isn't needed, so I've simplified the circuit plan to remove the unnecessary extra part. The older plan will still work just fine, too, it's just more complicated than it has to be.

Pre-built MOSFET booster

Thanks to the popularity of Arduinos and hobby robotics, you can buy pre-built boards that implement roughly the same circuit as the DIY MOSFET booster above.

There's no name-brand board or seller that I can point you to, unfortunately. The boards I've seen are sold on eBay and Aliexpress.com, so the sellers are small shops that come and go. If I gave you a link here, it would probably be gone by the time you tried it. So instead I'll tell you what to search for on those sites: try something like "Four channel MOSFET board".

Most boards like this will have a very similar design to the DIY MOSFET circuit shown above. The one big difference will be that the resistor between the signal input terminal (where you connect the KL25Z GPIO pin) and the optocoupler will typically be smaller than the one in the DIY circuit. That's because generic boards like this will always be intended for use with an Arduino, which has much higher power capacity on its GPIO pins than the KL25Z does.

So I recommend the following procedure:

- Set up the board as recommended by the manufacturer, but **add a 1K resistor** between the KL25Z GPIO pin and the input terminal on the board.
- Test it (with the Pinscape Config Tool's output tester, for example) to see if the output turns on as expected. If so, you're done!
- If it doesn't turn on, try removing the 1K resistor and replacing it with a 680 Ω resistor. Repeat the test. If it works, great; if not, drop to 470 Ω and try again.

The point of the extra resistor is to reduce the current on the GPIO port to a safe level for the KL25Z. The KL25Z has a maximum limit of **4mA** per port. These boards are usually designed to work at about 20mA, because the Arduino can tolerate the higher current.

If you have a multimeter, you can test the actual current being drawn on the port to verify that it's within the safe range, below 4mA. You can try resistors below 470 Ω if necessary, but if you do you should measure the current to verify that it's not too high.

Sainsmart booster

There's a non-USB version of the Sainsmart relay boards, designed mostly for Arduino users, that can be used with the KL25Z. Some people using the Pinscape software have successfully used that as a booster. However, I don't recommend it, because I've heard too many bad things about reliability. The relays reportedly have a short lifetime (weeks or months) when used in a pin cab. The transistor solutions above are much more reliable, and probably cheaper.

LED strip mini amplifier

If you search on eBay for "LED strip mini amplifier", you'll find products that look like this:



Some people have used these successfully with the KL25Z as a power booster circuit. However, I don't recommend doing this, because some of these devices are electrically incompatible with the KL25Z and can damage it if attached directly. It's difficult to tell whether a given mini-amp is safe to use with the KL25Z, so my advice is simply to avoid them entirely.

Testing and troubleshooting

The best way to test your connections is using the Pinscape Config Tool:

- Make sure all of the wiring is in place for the feedback device you want to test
- Run the Config Tool
- Click on the Output Tester button
- Find the port you want to test in the list
- For a PWM port, use the slider to test the brightness control. For a digital out port, use the ON/OFF button to test it.

The output tester sends commands directly to the KL25Z via the USB connection, so it bypasses other software layers (such as DOF and VPinMAME). If a port is working in the output tester, you can be confident that the hardware and wiring is all working correctly, and that any problems you're having with it in Visual Pinball or other programs are purely software configuration issues. If a port *isn't* working in the output tester, the problem is probably in the wiring or in the feedback device itself.

Voltmeter tests

You can do some simple testing with a voltmeter to confirm that the KL25Z is sending the right signal to the GPIO pins.

This test has to be done with the power on, so be careful! Make sure that you only touch the meter's probes to the pins you want to test, and that you don't short together adjacent pins. The pins are closely spaced, so you have to be really careful about this.

- Disconnect the output from the GPIO pin you want to test (including both the feedback device **and** the booster circuit)
- Power up the system
- Run the Pinscape Config Tool
- Click on the Output Tester button
- Set your meter to DC Volts
- Connect the meter's black probe to one of the KL25Z's ground (GND) pins, *or* to any convenient ground connection to your power supplies. (Remember that all of the grounds should be interconnected, as described in Chapter 45, Power Supplies for Feedback, so any of these points should be equivalent.)
- Connect the meter's red probe to the GPIO pin you want to test
- When the port is **off** in the output tester, it should read close to 0V (exception: an Active-Low port should read close to 3.3V)
- When the port is **on** in the output tester, it should read close to 3.3V (exception: an Active-Low port should read close to 0V)

If this test passes, your KL25Z is working correctly, and the software is configured properly so that the software port is connected to the physical GPIO pin. Any problems must be in the wiring to the booster circuit and/or to the feedback device.

If this test fails, either your KL25Z hardware is broken, or you have a problem with the software configuration. Check carefully that the pin you're looking at in the output tester matches the physical pin that you're testing on the KL25Z. Look very closely - make sure that you don't have the KL25Z flipped upside-down, for example, since that would make all of the pins in the mirror-image locations from where you

think they are.

DOF Setup

To set up your standalone Pinscape device with DOF, you of course have to install DOF on your PC first. See Chapter 46, DOF Setup for instructions.

Once the DOF software is installed, you use the DOF Config Tool to tell DOF that you have a Pinscape unit, and to tell it which KL25Z pins are attached to which output devices.

- Open the DOF Config Tool in your browser
- Click the My Account tab
- Set **Number of Pinscape devices** to 1 (or if you have more than one KL25Z running the Pinscape, select the appropriate number of devices instead)
- Set **Number of KL25Z devices** to **0** (see below if this seems confusing)
- Save changes
- Go to the Port Assignments page
- Select "Pinscape 1" in the Device drop list
- Go through the port list, assigning each port number in the DOF list to the device that you wired to that port on the KL25Z

The port list in the Port Assignments page uses the same port numbering (Port 1, Port 2, etc) that's shown in the Pinscape Config Tool output port list. This **isn't** the GPIO port name or physical pin number on the KL25Z. It's just the abstract port number from the output port assignment list.

To figure out what DOF's "Port 1" or "Port 2" means in terms of the physical GPIO pin on the KL25Z, you have to look at the output port list in the Pinscape Config Tool's Settings page. In the output list, find the same port number shown in DOF - if you're looking for DOF's "Port 1", you want the first row, #1, in the Pinscape output list. Trace across the row to find the KL25Z port name and pin. If you want to see a picture of where that pin is physically located on the KL25Z, click the pin name in the row - that will pop up the pin selector, which will show the pin location highlighted on a picture of the KL25Z.

Using two (or more) KL25Zs

If you like the idea of using a standalone KL25Z as your output controller, but you need more ports, there's an easy solution: add a second KL25Z. Or even a third or fourth. Additional units give you more output ports - and on secondary units, you don't even have to share GPIO pins with other functions like buttons and plungers. You only need those input pins on the first KL25Z unit, so nearly all of the pins on secondary units can be reassigned as outputs.

The Pinscape software is set up so that you can have up to 16 separate KL25Z units in your system. The only additional setup work required to add a second KL25Z is to give it a unique ID, using the Pinscape Config Tool:

- Detach all of your other KL25Z units
- Attach the new one you want to program with a new ID
- Run the Pinscape Config tool
- Go to the Settings page for the unit

- In the **USB ID** section, set the ID to LedWiz Unit 9
- In the **Pinscape ID** section, set the ID to Pinscape Unit 2
- In the **Joystick** section, un-check the "Enable joystick input" box

Repeat this process for each additional KL25Z, advancing to the next LedWiz Unit number and Pinscape Unit number for each additional device.

(The USB ID doesn't actually have to be an LedWiz unit number at all, and doesn't have to be in this exact order. That's just the recommended order for the widest software compatibility. You should, however, use Pinscape unit numbers 1, 2, and so on, in that order. The DOF Config Tool is difficult to set up otherwise.)

Why disable the joystick input? Because you want to make sure that only one of the KL25Z units is sending accelerometer (nudge) readings to the PC. Pinscape sends those readings by default, so you have to turn off the readings in all of your secondary units to avoid confusing the PC with a barrage of different information from different units.

DOF setup: To set up a second Pinscape unit with DOF:

- Open the DOF Config Tool in your browser
- Click the My Account tab
- Change the "Number of Pinscape devices" to the appropriate number
- Save changes
- Go the Port Assignments page
- Select "Pinscape 2" in the Device drop list
- Set up the ports for the devices attached to the second KL25Z
- Repeat for additional units

Is it a "Pinscape device" or "KL25Z device" or both?

The online DOF Config Tool has a confusing bit of terminology in the device setup section on the "My Account" page. In the list of devices, you'll find separate entries for "Number of Pinscape devices" and "Number of KL25Z devices".

For Pinscape boards, use **only** the **Pinscape devices** option. Always leave "Number of KL25Z devices" set to zero (0). This applies whether you're using the expansion boards or a standalone KL25Z.

This is confusing because the Pinscape software does happen to run physically on a KL25Z board, so it might seem like you should enter the same number for both line items. Don't. Pretend that you've never heard of a KL25Z and that you have zero of them.

The Config Tool has the "KL25Z devices" line item for historical reasons that date back to the first version of the Pinscape software, when it was limited to 32 output ports. It should more properly be titled "Number of Pinscape v1 devices", because that's what it really means. In any case, just ignore it and leave it set to zero.

50. LedWiz Setup

The LedWiz has long been the most popular output controller for virtual cab builders. It was originally created for video game cabinets, not for pin cabs, but the two have a lot in common: they both use a PC to run their software, and they both use arcade controls built into a custom enclosure. Video game cabs use the LedWiz mostly to control button lamps and other cabinet lights. But a device like the LedWiz that can turn lights on and off can really turn just about anything on and off, so it filled the need when early pin cab builders wanted to install mechanical force feedback devices in addition to lights.

The LedWiz has a lot of virtues. It's inexpensive, it has a "driverless" software design that makes it practically effortless to set up in Windows, it has lots of output ports (32), and it has the ability to control the brightness on all ports via PWM. Its only serious limitation is power handling.

Power limits and boosters

The LedWiz has fairly low power limits, because of its legacy as a video game device. In a video game cab, the only "feedback" devices most people install are small light bulbs and LEDs. The LedWiz's designers chose the power limits accordingly.

The exact limit is 500mA per output port. (To be really technical, there's also a limit on the combined power across groups of ports, but we can mostly ignore that in a pin cab; that only applies if all of the ports are switched on for long periods at the same time, which essentially never happens in a pin cab.)

500mA is plenty for the light bulbs and LEDs that the LedWiz's designers had in mind. But it's not enough for some of the mechanical devices we use in pin cabs. Many types of solenoids, contactors, coils, and motors can greatly exceed this. Some high-power LEDs can exceed it, too. So whatever you're connecting, be sure you know how much current it uses, and be sure it's safely under the 500mA limit. If you can't find information on a device's power usage in its instructions or spec sheet, you can measure it with a multimeter set to Current (Amps) mode.

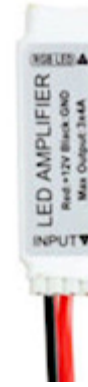
What should you do if a device exceeds the 500mA limit? Well, for starters, **don't** connect it directly to the LedWiz. If you do, you'll likely destroy at least the one LedWiz port that you connected the device to, and possibly a whole group of eight ports (because each group of eight ports is controlled by a single chip on the LedWiz, and an overload on one port can destroy the whole chip).

What you need in this case is a "booster" circuit of some kind. A booster is something that increases the power handling on a port, which it accomplishes by adding a second electronic switch between the LedWiz and your high-power feedback device. The LedWiz controls the booster's second switch, and the second switch controls the feedback device.

Here are the main types of boosters you can use:

- A booster board from zebboards.com. Zeb's sells several boards that connect to an LedWiz. These use high-power transistor switches, so they're compatible with PWM brightness control. This is the easiest option, because Zeb's products are built specifically for virtual pin cab use, so you won't have to improvise anything to fit them into your system.
- Your own DIY booster circuit using a MOSFET or other transistor. We provide a simple circuit design below. It's easy to build, and it's powerful enough for just about any type of pin cab device.

- An LED light strip amplifier from eBay. Search for "Mini LED amplifier", and look for products similar to the one pictured at right; they go for about \$1 apiece if you buy a lot of 5 or so. These use transistor switches based on the same principles as Zeb's boosters or the DIY booster circuits described above. These are designed to be used with RGB light strips, so each unit has three independent channels of amplification, so you'd need 11 of them to boost all 32 ports on an LedWiz. Each channel can typically handle 4A, which is enough for most pin cab devices. The main drawback with these is that it you'll have to improvise a bit to wire them, since they're physically designed to mate with the common LED strips.



- Individual relays. A relay is a mechanical switch controlled by an electromagnet. Normally, the switch is disconnected ("open"). When the magnet is energized, the switch "closes", connecting the attached wires. To use a relay as an LedWiz booster, you connect the relay coil to the LedWiz as though *it* were the feedback device; that lets the LedWiz turn the relay switch on and off according to the port status. You then connect the actual feedback device to the relay switch. When the LedWiz turns the port on, it energizes the relay, closing the switch and connecting power to your feedback devices. This is fairly easy to set up, and you can find relays that can handle enough power for large devices. The downside is that a relay can't handle PWM for brightness control, so it's not good for lights, and also doesn't work for motors if you want variable speed control.
- A SainSmart relay board (the non-USB type). SainSmart sells boards with different numbers of relays (from 1 to 16) that can be controlled by a device like the LedWiz. This is the same idea as using an individual relay as described above, but the SainSmart boards make it a little more convenient if you want to boost multiple outputs this way. As with individual relays, the SainSmart relays can't handle PWM, so they're not great for lights or adjustable-speed motors.

Note that SainSmart sells two completely different kinds of relay boards: USB and non-USB. For an LedWiz booster, you want the **non**-USB type, also known as an "Arduino module", since they're sold primarily to Arduino users. Don't get the USB type for this use case, since that's a standalone controller that you'd use *instead of* an LedWiz.

What about the "H-bridge hack" I've read about for motors?

Back in the early days of pin cabs, someone came up with the idea of hacking into the LedWiz's internal wiring to connect a special type of booster circuit known as an H-bridge. The idea came from the Arduino robotics world, where H-bridges are often used to control robot wheel motors from Arduino GPIO ports. Someone in the pin cab world must have read about it and thought that "H-bridge" and "motor" just naturally go together, and the idea made its way into the forums and got repeated a lot. But you actually don't need an H-bridge to control any of the common virtual pinball motor devices.

The special feature of an H-bridge is that it can be boost both positive and negative input voltages. That's useful with robot motors because it lets you run the motor bidirectionally. But in a pin cab, we have no need to run our motors in reverse. The motor devices we use - shakers, gear motors, fans, beacons - only need to run in one direction.

So in a pin cab, **you don't need an H-bridge** for your motor devices. You just need one of the general-purpose boosters we cover here. The only special requirement is

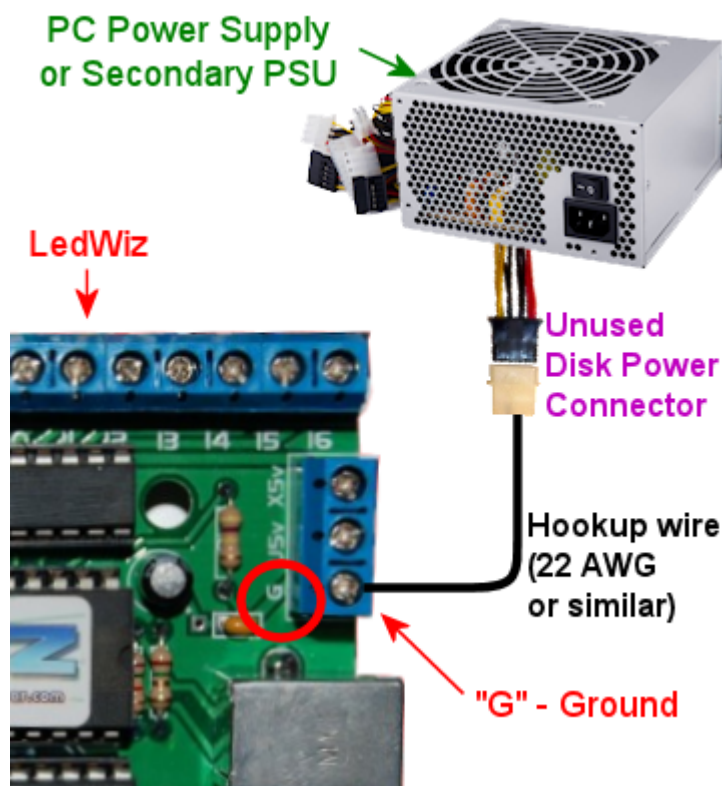
to make sure the booster you choose has a high enough current limit for your motor. Shaker motors and windshield wiper motors (if that's what you're using for your gear motor) run at about 4A, while fan motors and beacons are typically more like 1A to 2A. The generic DIY MOSFET circuit described later in this section can easily handle any of those loads.

Note that you could use an H-bridge if you really wanted to, but there's no benefit, and they're more complex and expensive to set up than a standard booster circuit. And the H-bridge "hack" for the LedWiz is actively harmful. The "hack" requires you to modify your LedWiz by soldering a wire onto a tiny pin on one of the LedWiz's surface-mount IC chips. It's a difficult soldering job because of the small pins involved, and it runs the risk of damaging the device. So I'd definitely ignore the whole H-bridge idea and use one of the simpler booster options instead.

Connecting the LedWiz to the PC

Connecting the LedWiz to the PC requires two bits of wiring.

First, you'll need some insulated hookup wire. (I'd recommend 22 AWG, but anywhere from 16 AWG to 24 AWG is fine.) Connect one end to the External Ground terminal on the LedWiz, which is labeled **G** (for Ground) and should be near the USB connector. Connect the other end of the wire to an unused disk power connector on your PC power supply **or** a secondary ATX power supply. Be sure to connect to one of the **black** wires in the disk connector.



*Connecting the LedWiz ground terminal. Use ordinary hookup wire (around 22 AWG) to connect the **G** terminal on the LedWiz to the black wire in an unused disk power connector on your power supply. The LedWiz comes in a couple of versions, so yours might not look exactly like the photo, but there should still be a similar **G** terminal.*

For more on connecting power wires to the PC power supply, including how to build your own connectors, see Chapter 45, Power Supplies for Feedback.

Second, plug a USB cable into the USB connector on the LedWiz, and plug the other end into a free USB port on your PC.

Positive power supply connections

Some versions of the LedWiz have a set of terminals for connecting the positive (+) voltage from your power supply. **You should simply ignore these and leave them unconnected.** These inputs are unnecessary as long as you use your own diodes for any attached coils or motors, and they complicate the wiring if you do use them. It's best to use your own diodes and ignore these inputs.

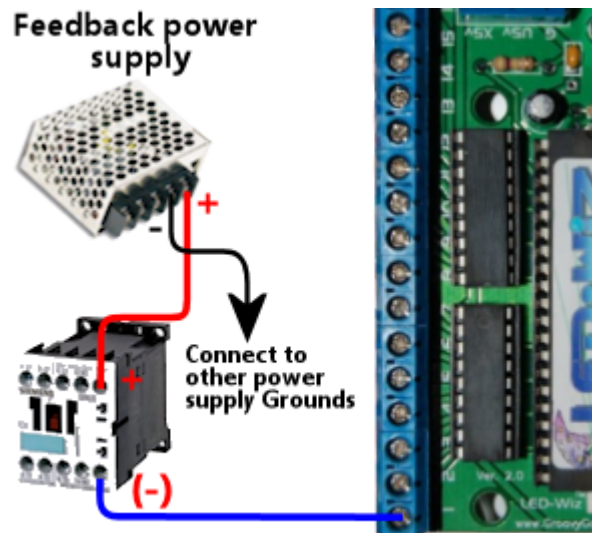
The function these inputs serve is to connect some internal diodes within the LedWiz chips across the output port terminals. The diodes are meant to provide a degree of protection against power spikes from inductive devices. That protection is important, but you can get better protection by installing your own external diodes, as explained in Chapter 53, Coil Diodes. Doing that will make the internal diodes redundant, allowing you to leave the LedWiz voltage inputs unconnected.

Basic device wiring plan

Before connecting anything to the LedWiz, please make sure its power draw is safely below the LedWiz's limit of 500mA per output. Any device that draws more power requires a booster circuit; see Power limits and boosters above. Attaching a device that draws too much power can damage or destroy the LedWiz.

The basic feedback device wiring plan for an LedWiz is pretty simple:

- All of the "-" or "Ground" connections from all of your power supplies should be connected together (see Chapter 45, Power Supplies for Feedback)
- Connect the "+" terminal from the power supply directly to the "+" terminal on the feedback device
- Connect the "-" terminal on the feedback device to one of the numbered LedWiz terminals



The LedWiz acts like an electronic switch that connects and disconnects the "-" side of a device to the power supply. The LedWiz can **only** switch the "-" side, so it's important to wire the polarities exactly as described above.

Some feedback devices aren't polarized, meaning they don't care which power input is "+" and which is "-". This is true for most coil devices, like solenoids and relays, as well as for incandescent lamps. If the device doesn't have "+" and "-" markings for its power terminals, simply follow the same wiring plan shown above, but you can attach the wires to the feedback device itself in either order.

Some devices have special requirements beyond the basic diagram shown above. For example, anything with a coil needs a diode (see Chapter 53, Coil Diodes), and most LEDs require current-limiting resistors (see Chapter 56, Flashers and Strokes). We go into more detail about each of the popular pin cab devices in Feedback Devices.

Use protective diodes if necessary

Most mechanical devices **require** protective diodes. These are required for anything with a coil: solenoids, contactors, replay knockers, motors.

See Chapter 53, Coil Diodes for details on what type of diodes to use and how to install them.

Diodes are critical! Coils and motors can damage your PC motherboard and other components if diodes aren't used.

Fuses

Many cab builders include a fuse in each LedWiz output circuit. This isn't strictly necessary, but I think it's a good idea with the LedWiz because of its low power limits. Fuses can help protect the LedWiz from overloads caused by wiring faults and device malfunctions, reducing the chance that the LedWiz will be damaged if something like that goes wrong. See Chapter 84, Fuses for details.

Windows setup

The LedWiz requires absolutely no Windows hardware drivers. It's completely plug-and-play.

Visual Pinball and PinballX access the LedWiz through DOF, so you'll need to set up DOF as described below.

LedWiz.dll

Some older software, such as Future Pinball, accesses the LedWiz through a "DLL" file (a Windows "dynamic link library") called ledwiz.dll. You can download this from the manufacturer's Web site (GroovyGameGear.com), but I recommend using my upgraded, open-source version instead. You can find it here: DOF R3 & LEDWIZ.DLL updates. My version works more reliably than the official DLL, because it has special logic to work around some recently discovered hardware bugs on the LedWiz itself.

LedWiz.dll should generally be installed in the same directory as each program that uses it (e.g., Future Pinball). You might need to install multiple copies if you're using multiple programs that use the DLL.

You **don't** need ledwiz.dll for Visual Pinball, DOF, or PinballX.

DOF setup

DOF is required to use your LedWiz with Visual Pinball and PinballX. If you haven't

already installed DOF on your system, follow the instructions in Chapter 46, DOF Setup.

Once you have the DOF software installed on your PC, follow these steps:

- Point your browser to the DOF config tool
- Click the My Account tab
- Find "Number of LedWiz Devices" in the list, and set it to "1" (or to the number of LedWiz units you've installed, if more than one)
- Click Save Settings
- Click the Port Assignments tab
- In the "Device" drop-down, select "LedWiz 1"
- Go through the port list (Port 1 through Port 32), and select the type of device that you've physically wired to each output port. The port numbers in the list correspond to the numbers printed on the LedWiz board next to the screw terminals for the ports.
- Click Save Config
- If you have more than one LedWiz, repeat the process for each device by selecting the devices one at a time in the "Device" drop-down at the top of the page
- When done, click Save Config, then click Generate Config to download your updated DOF config files
- Unzip the downloaded files into your DOF install folder

All of this is described in the Chapter 46, DOF Setup section as well.

Power boosters

Let's look at how to use the various booster options with an LedWiz, in case you need more power for devices that exceed the limit of 500mA per output.

Zeb's boards boosters

If you're using a booster board from Zeb's boards, see the wiring plans included in the instructions that come with the board.

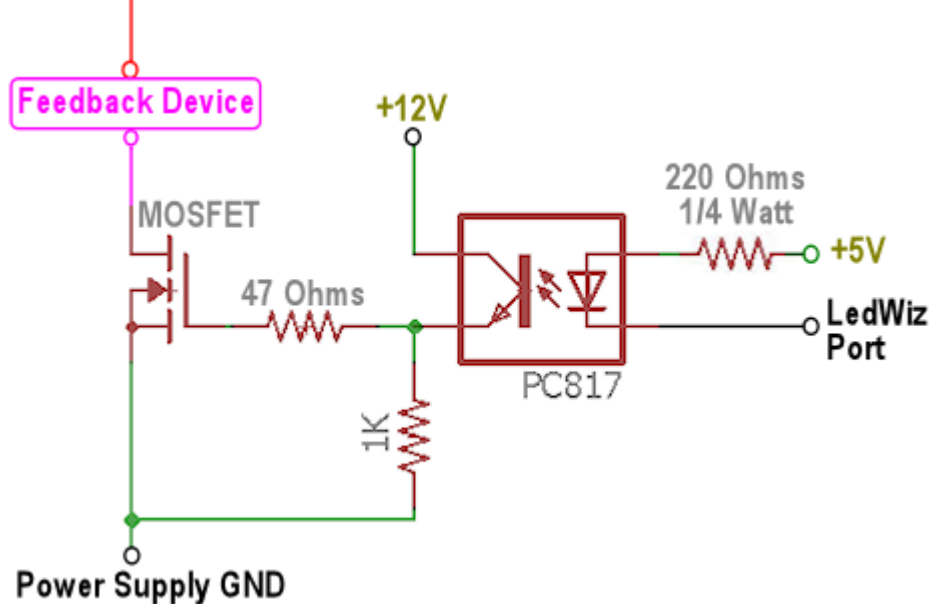
DIY booster circuit

Here's a circuit plan for a generic booster circuit that you can build yourself with a few components. This is basically the same circuit used in the Pinscape expansion boards. It's a great general-purpose, high-power booster circuit that can drive just about anything in a pin cab, from motors and solenoids to flashers and button lamps. This circuit is fully compatible with the LedWiz's PWM dimming control.

The power limit for the attached device is determined by the type of MOSFET you choose. The MOSFETs listed below can all handle upwards of 10 Amps, which is plenty for just about any pin cab device. (Shaker and gear motors typically run at around 3A to 4A, and large pinball solenoids like knocker coils need about 4A. All of the other common devices have much lower power needs.)

Here's the circuit plan:

Power Supply (+)
○



If you need help decoding the schematic, see A Crash Course in Electronics.

The resistors can all be filled with any ordinary resistor of the specified "Ohms" size, except for the 220 Ω resistor, which should have a power rating of 1/4 Watt or higher. (You can always use a higher wattage than specified with resistors.)

Which MOSFET to use? Here's a list of parts I've tried that work well:

- BUK7575-55A
- FQP13N06L
- FQP30N06L

But lots of other MOSFETs will work just as well. Any N-channel type sold by an Arduino or robotics company will probably be suitable, since robotics projects often use these parts exactly the same way we do (and for the same reasons). If you want to cast a wider net by looking on Mouser, the basic type of part you need is an N-channel enhancement-mode MOSFET - but that turns up about 8,000 matches on Mouser, so here are some more specific characteristics to look for:

- Low "on" resistance ($R_{DS(on)}$), below 1 Ω (preferably something like 100m Ω)
- Drain-source voltage (V_{DSS}) sufficient for your feedback device power supply, preferably above 40V
- Continuous drain current (I_D) sufficient for your feedback device's needs, preferably above 10A
- Through-hole package (for easier soldering)

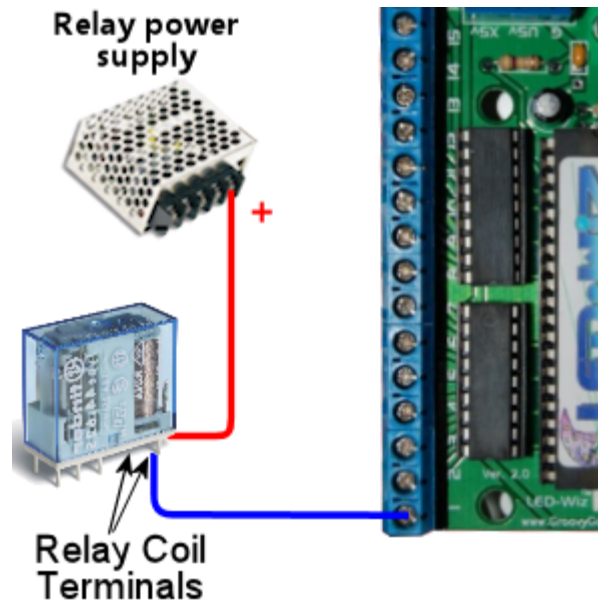
Here's a Mouser search for those characteristics. This still matched about 1,400 parts when I tried it, so it doesn't exactly narrow things down to a trivial selection, but I'd sort by price, pick one of the cheaper ones, and scan the data sheet to make sure it looks like a suitable part for logic applications.

[www.mouser.com/Semiconductors/Discrete-Semiconductors/Transistors/MOSFET/_/N-ax1sfZ1yzvvqx?](http://www.mouser.com/Semiconductors/Discrete-Semiconductors/Transistors/MOSFET/_/N-ax1sfZ1yzvvqx?P=1z0y3zrZ1yiaumvZ1z0z63xZ1z0y4ci&Rl=ax1sfZgjdhp3Z1yw78ezZ1ypyijjSGgjdhozZ1yw76gfZ1yvi0)

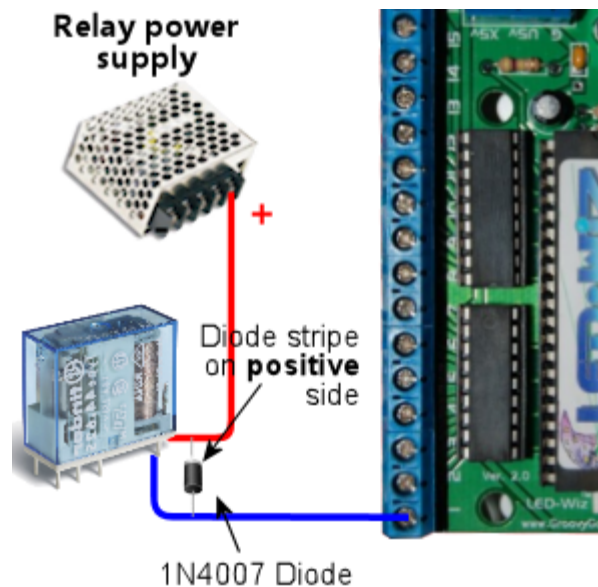
[P=1z0y3zrZ1yiaumvZ1z0z63xZ1z0y4ci&Rl=ax1sfZgjdhp3Z1yw78ezZ1ypyijjSGgjdhozZ1yw76gfZ1yvi0](http://www.mouser.com/Semiconductors/Discrete-Semiconductors/Transistors/MOSFET/_/N-ax1sfZ1yzvvqx?P=1z0y3zrZ1yiaumvZ1z0z63xZ1z0y4ci&Rl=ax1sfZgjdhp3Z1yw78ezZ1ypyijjSGgjdhozZ1yw76gfZ1yvi0)

Individual relay booster

The idea here is to connect the relay coil to the LedWiz as though *it* were the feedback device. So you connect the coil exactly as you would any other device.

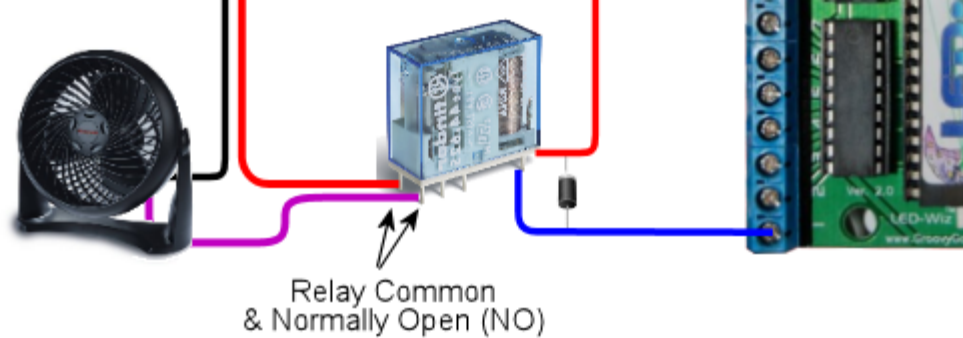


You must also connect a protective diode across the coil's terminals, to suppress the voltage spike that the coil will produce when it switches off. A 1N4007 diode is a good general-purpose diode that you can use for this. See Chapter 53, Coil Diodes for full details.



Then you connect the actual feedback device to the relay's switch terminals. Most relays are of the "double-throw" type, meaning they have three switch terminals: common, normally open (NO), and normally closed (NC). You connect the device to the common and normally open terminals.

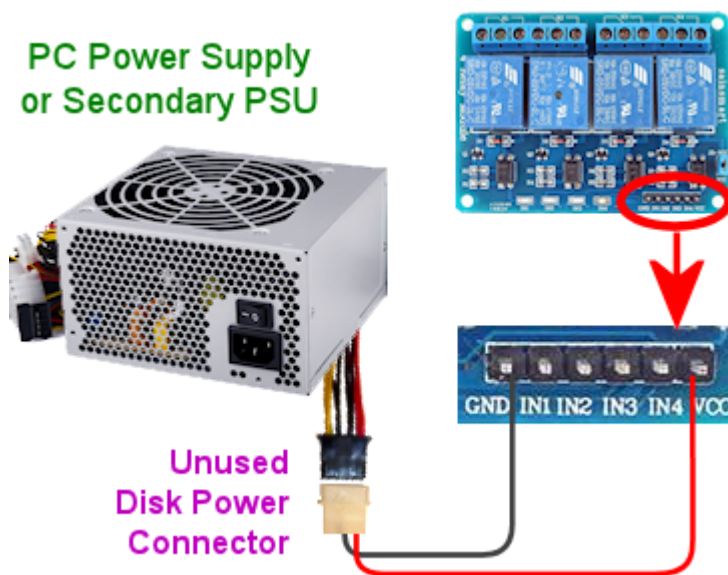




When shopping for a relay, pay attention to two important specifications. First, make sure its **coil current** is safely below the LedWiz's 500mA limit. Most small relays are well within this, but some larger relays require more current. Second, make sure that its switching load limits are above the voltage and current required by the feedback device you intend to attach. Pay particular attention the **VDC** (Volts DC) number and make sure it's above the power supply voltage for the feedback device. Maximum DC switching voltages are usually much lower for relays than their AC switching voltages because DC voltage creates arcs more easily in the switch contacts, so you might see a relay rated for 250VAC but only 30VDC.

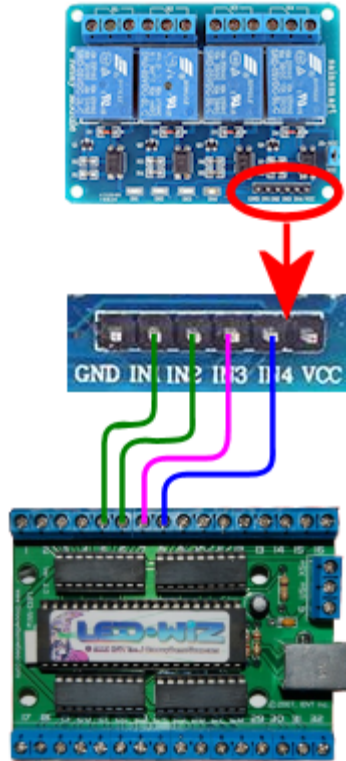
SainSmart relay board as a booster

Start by connecting the relay board to the power supply. Find the pins labeled **GND** and **VCC**, and connect these to your PC power supply (or secondary ATX power supply), using an unused disk connector. Connect GND to the power supply black wire. The SainSmart boards come in 5V and 12V versions. If you have a 5V version, connect VCC to the red power supply wire; for the 12V version, connect VCC to the yellow power supply wire.



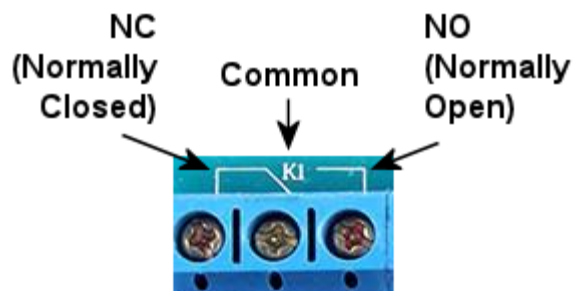
SainSmart relay board power hookups. Connect the SainSmart GND pin to the black wire in a disk power cable from the PC power supply. Connect the SainSmart VCC pin to the red wire if you have a 5V board, or the yellow wire if it's a 12V board.

Next, connect the relay board's input pins to the LedWiz output ports. Find the pins labeled **IN1**, **IN2**, etc. These are the inputs. **IN1** is the input for the first relay, usually labeled **K1**, and the other inputs correspond to the other relays the same way. Connect a wire between each **INx** pin and an LedWiz output port terminal.



Note that the choice of four LedWiz ports shown in the diagram is arbitrary, purely for the sake of illustration. You can connect any other ports instead, and they don't have to be adjacent as shown.

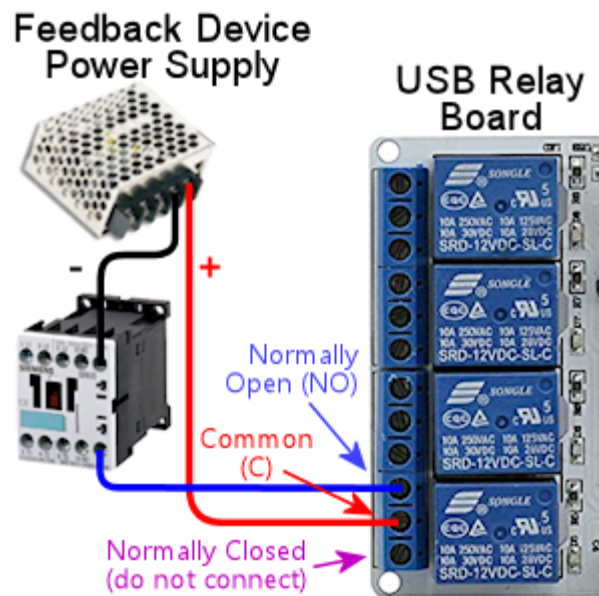
Finally, connect the output devices. Each relay has three terminals, and you need to identify the **Common** and **Normally Open** connections. The markings vary by board. You might see the legends **NC**, **C**, and **NO** printed on the board next to the terminals. These stand for Normally Closed, Common, and Normally Open. You might instead see a little line diagram like this:



If you can't find any markings, look for the set of three terminals nearest the relay. The Common is almost always the center terminal, and the Normally Open is usually the top terminal when the relay is oriented like this:

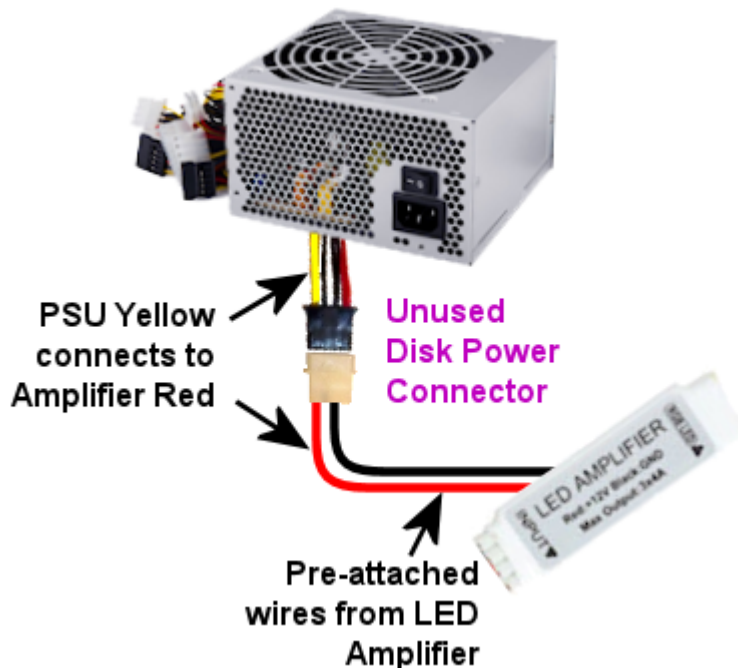


Once you identify the Common and Normally Open terminals for the relay, connect your feedback device like this:



LED amplifier as a booster

First, connect the LED amplifier to the power supply. You need a 12V power supply for this; you can use a PC ATX power supply, preferably a secondary unit rather than the one that's powering your PC motherboard. Use the **yellow** and **black** wires for 12V. The amp should have a pair of wires attached, one red and one black. Red connects to +12V, black connects to the power supply ground.

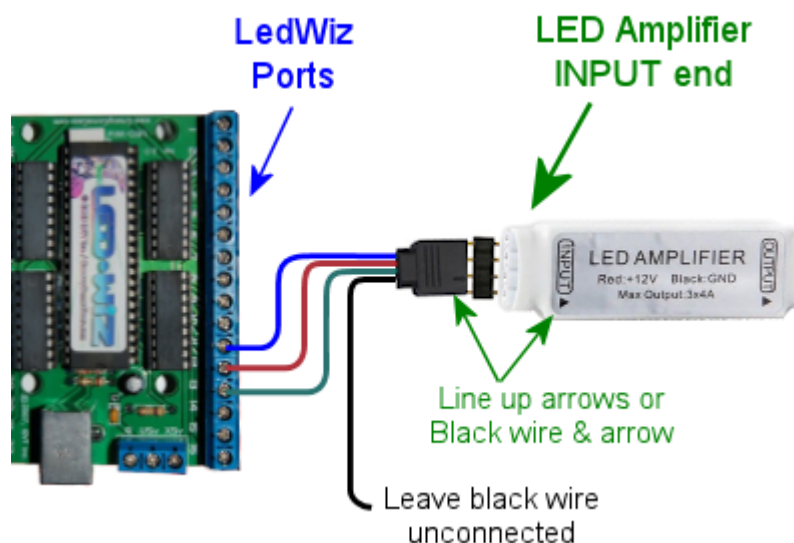


Next, connect the LED amplifier to the LedWiz output ports. Run a wire from each of the three **Input** pins on the amplifier to a port terminal on the LedWiz. **Don't connect the amplifier pin labeled "+" or ►.**

The inputs and outputs on these amplifiers are designed to mate with special 4-pin connectors. They're basically just thick wires, so you might be able to improvise something, but the easiest approach is to buy some of the special connectors. These can be found on eBay: search for "LED male connector". The easiest ones to work with, in my opinion, are this type:

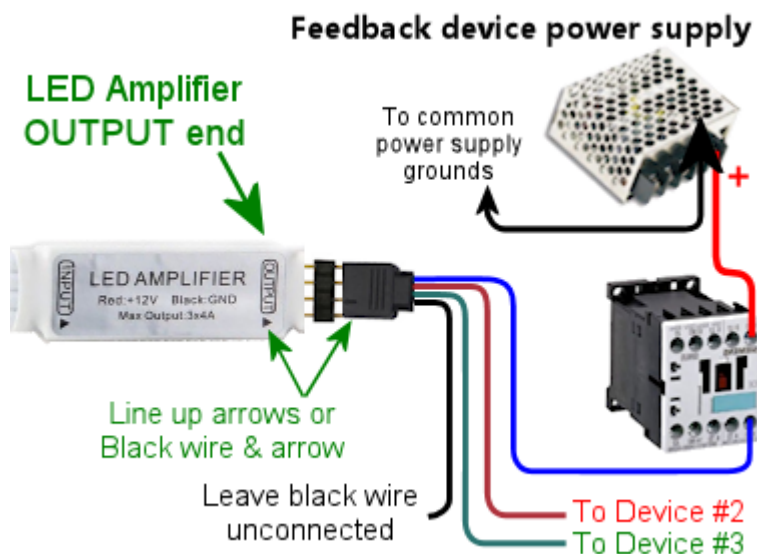


I like these because they use plain wires between the connectors, so you can cut off the LED strip connector (the white connector in the type pictures above) and be left with four wires that you can easily attach to other types of terminals, such as the LedWiz terminals. Using that type of connector, the connection to the LedWiz looks like this:



Note that the specific ports shown in the diagram are arbitrary. You can connect any three ports to each amp, and they don't have to be adjacent. But I'd try to keep things organized cleanly anyway, for your own, so that you can follow your wiring more easily when you come back to it later.

Finally, connect the feedback device to the amplifier. Wire the feedback device's "+" terminal to the power supply "+", and wire the device's "-" terminal to one of the three **Output** pins on the amplifier. The output pins are the same form factor as the input pins, so you can use the same kind of connector described above to connect these.



Note that each output pin on the amplifier corresponds to the input pin across the case from it. You can use that to determine which LedWiz port controls which output.

Using multiple LedWiz's

You can use more than one LedWiz in the same system, but there's a slight catch: when you buy the second or third or Nth LedWiz, you have to special-order it as the second or third or Nth device.

The reason this is necessary is that each LedWiz unit has a built-in hardware ID that it uses to communicate with Windows over the USB wire. The LedWiz documentation calls this the "device number" or "unit number". This can only be set at the factory, so you have to specifically order each device with a different unit number. If you don't specifically ask, the unit number is usually "1". So if you want to connect a second LedWiz to your system, you have to make a special request for "unit number 2" when you buy the second one. If you want to connect three, ask for "unit number 3" when you buy the third one. And so on.

If you buy directly from the manufacturer, GroovyGameGear.com, look for the "Choose device number" area on the ordering page, and enter the device number you need.

If you buy from eBay or another seller, you'll be stuck with whatever unit number was set at the factory. That's completely fine if you only need one LedWiz in your whole system. But if you want to connect more than one, you'll have to order each unit with a different device number, so you should order from someone who lets you specify this when ordering.

51. SainSmart Relay Board Setup

The SainSmart USB relay board is an inexpensive output controller option, and it's fairly easy to set up electrically. It's somewhat more trouble to set up the software for these devices, though.

Warning: DOF currently only supports the **8-channel** version of the Sainsmart USB board. Sainsmart makes the USB boards with different numbers of relays, from 4 to 16 channels, but DOF is only programmed to recognize the 8-channel version. Other types won't work with DOF.

The big advantage of the SainSmart boards over the LedWiz and PacLed controllers is that SainSmart boards can directly control high-power feedback devices like motors and solenoids, because they use relays instead of transistors for switching. The LedWiz and PacLed are designed primarily for switching low-power devices like LEDs, and their power limits are too low for most motors and solenoids. The SainSmart relays, in contrast, can handle just about anything commonly used in a pin cab. In addition, a lot of pin cab builders prefer relay switches over the transistor-based boards just because they're a more familiar technology.

The SainSmart USB boards come in 4-relay and 8-relay versions. You can use more than one board in your system, so you can have as many ports as you need simply by adding more boards.

The SainSmart boards are most attractive if you only plan on a relatively small set of feedback devices, and all of them are mechanical. If that describes your needs, these boards are a good option. However, the SainSmart boards aren't the best choice if you want a decked-out cab with all of the popular toys. For one thing, they're not great if you need a lot of ports, because even though you can keep adding boards as needed, the individual boards take up a lot of space. The solid-state controllers pack more ports into less space. The other thing that makes SainSmart boards less than ideal for a complex cab is that they don't work very well with LEDs, because they can't vary the brightness - they're strictly on/off switches. If you're planning to use flasher LEDs or other lighting devices, you'll really want a solid-state controller (such as LedWiz, PacLed, or Pinscape), since those provide brightness control in addition to on/off switching.

Warning: there seem to be some no-brand relay boards out there that look very much like Sainsmart boards, with the same blue relays laid out the same way, but which aren't compatible at the software level. I'd avoid anything like this that isn't sold with clear Sainsmart branding, because the no-brand units probably won't work with DOF and the other pinball software.

USB vs non-USB boards

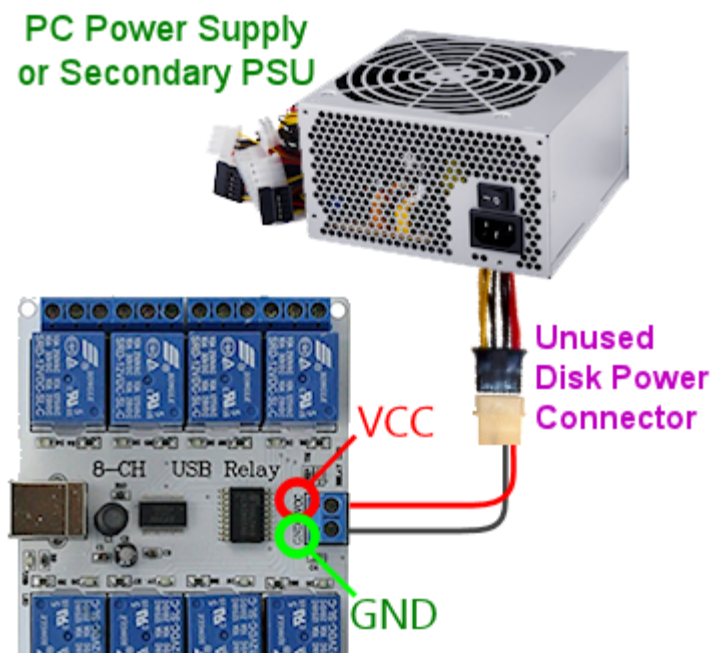
When you go shopping for these boards, be aware that SainSmart sells two completely different kinds of relay boards: USB and non-USB.

- The USB type has a USB port for connecting to the PC. This is the type we're talking about in this chapter, and it's the type you should buy if you *don't* want to have to combine it with a separate device like an LedWiz.
- The non-USB type doesn't have any ports for connecting to a PC. Instead, it's designed to be used with a microcontroller device like an Arduino, so it has individual control terminals for the relays that the Arduino connects to. Don't buy this type if you're looking for something standalone. However, this type *can* be used as a booster for the LedWiz (more about that in Chapter 50, LedWiz Setup).

Connecting to the PC

There are two steps required to connect the board to the PC.

First, connect the board's power terminals to your PC's power supply, as shown in the diagram below. Connect the GND terminal to the PC power supply ground (the black wire in any of the disk connector cables coming out of the PSU), and connect the VCC terminal to +5V (the red wire in any disk connector).



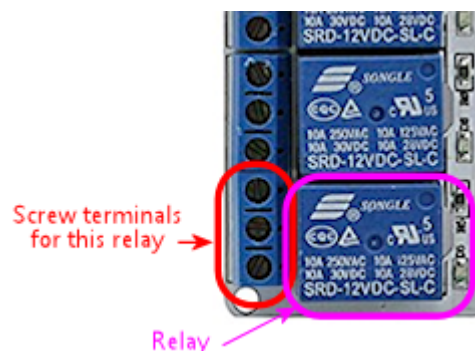
For more on connecting power wires to the PC power supply, including how to build your own connectors, see Chapter 45, Power Supplies for Feedback.

Second, connect the relay board to the PC via USB. Plug a USB cable into the connector on the board, and plug the other end into an available USB port on your PC/motherboard.

These boards require software drivers to be installed on the PC. Follow the instructions that come with the board.

Basic wiring plan

Each relay on the board can control one feedback device (for example, one relay can control your shaker motor, another relay can control your left flipper device, etc). You can see on the board that each relay has a group of three screw terminals next to it:

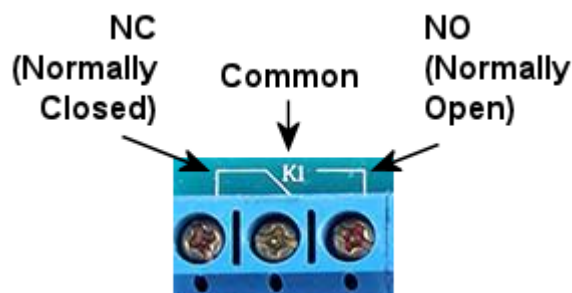


A relay is simply a mechanical switch, a lot like a regular pushbutton switch, except

that it's controlled electronically by a little solenoid coil inside the relay that moves the switch lever. The relay turns the attached device on and off by connecting and disconnecting the power to the device through the internal switch. It's just like turning on the lights in your house with a wall switch, except that the solenoid inside the relay is what moves the switch lever up and down.

The relay's switch section has three terminals, not just two, because the relay is a "double-throw" switch. That means that the switch can connect the center terminal (the common connection) to either of the two outside terminals. One terminal is the "normally closed", often abbreviated to "NC", meaning that it's connected to the center terminal any time the relay is OFF ("closed" means "connected" when we're talking about a switch, and "normally" means "when the relay is OFF"). The other terminal is "normally open", or "NO", meaning that it's not connected to anything at all when the relay is OFF ("open" means "not connected"), but it is connected to the center terminal when the relay is switched ON.

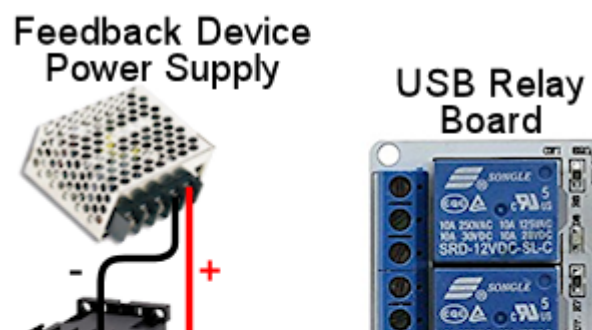
Your wiring for the device should be connected to the **center** (or common) and **normally open** terminals. Check the documentation that came with your board to identify these. You can also usually tell by markings on the board, but these vary. You might see the legends **NC**, **C**, and **NO** printed on the board next to the terminals. These stand for Normally Closed, Common, and Normally Open. You might instead see a little line diagram like this:

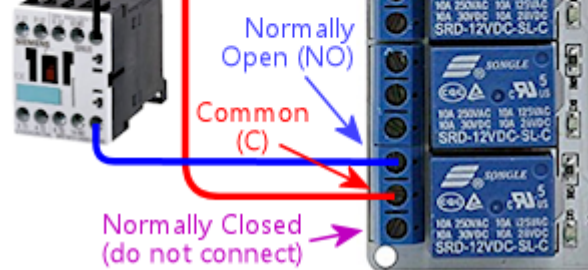


If you can't find any markings, look for the set of three terminals nearest the relay. The Common is almost always the center terminal, and the Normally Open is usually the top terminal when the relay is oriented like this:



Once you identify the Common and Normally Open terminals for the relay, connect your feedback device like this:





Follow this same plan for any type of feedback device. If the device you're connecting has specifically marked "+" and "-" terminals, be sure to connect the "-" terminal on the device to the "-" connection on the power supply.

Some types of devices have special requirements beyond the basic diagram shown above. For example, anything with a coil needs a diode (see Chapter 53, Coil Diodes), and many types of LEDs require current-limiting resistors (see Chapter 56, Flashers and Strobes). We go into much more detail about many of the popular pin cab devices in Feedback Devices.

Should I wire (+) or (-) to the relay?

In the diagram above, we showed the wiring with the (+) connection from the power supply going to the relay switch, and the (-) connection going directly to the feedback device.

If you look at the "general purpose" wiring diagram for other controllers in Chapter 47, Feedback Device Wiring, though, you'll see the opposite setup, with the (-) connection going to the controller port, and the (+) power supply connection going directly to the feedback device. We show the same polarity in our diagrams for the individual device types as well.

So why did we show this as different for Sainsmart? It turns out that you can do it either way with Sainsmart boards. For the solid-state boards, you *must* use the polarity shown in Chapter 47, Feedback Device Wiring, because the solid-state boards use transistors, which only allow current to flow in one direction. The Sainsmart uses mechanical relay switches, though, which don't care about the direction of the current. That makes them more flexible, letting you hook up the power wiring in either direction.

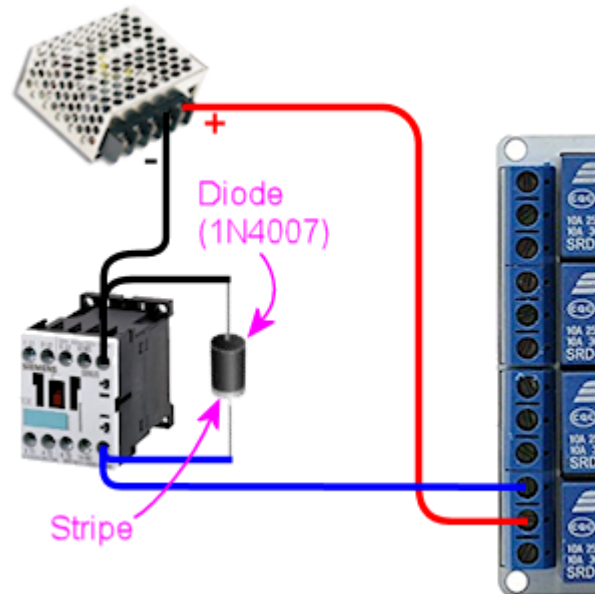
I know that I'm creating a little confusion here by showing it both ways, but there's actually a reason for it. Maybe not a good enough reason to justify the confusion, but a reason nonetheless. Specifically, given the choice of polarities that we get with the Sainsmart, I'd rather put the relay switch on the positive side of the circuit, because it's safer. When you put the switch on the positive side, as shown in the diagram above, the power supply wiring to the device itself is "dead" when the device is off - there's no voltage going to it. That eliminates any shock risk if you should accidentally touch any of the wires to the device. When you wire things the other way, as we must do for the solid-state controllers, the positive wire is always "live", even when the device is off, so it always runs the risk of shock. For 12V or even 24V devices, the voltage in the live wire isn't that dangerous, but it really is a safety hazard for a 50V device like a knocker coil.

So my apologies for the conflicting diagrams! I hope that this explanation clears it up, but if you still find it confusing, here's some good news: you can follow *either* diagram with your Sainsmart wiring (the special Sainsmart diagram above or the "general purpose" diagrams shown everywhere else), and it'll work equally well either way.

Use protective diodes

Most mechanical devices **require** protective diodes. These are required for anything with a coil: solenoids, contactors, replay knockers, motors.

Here's the basic wiring plan:



See Chapter 53, Coil Diodes for details on what type of diodes to use and more details about how to install them.

Diodes are critical! Failing to include one where needed can result in damage to your computer motherboard, power supplies, or other equipment in your pin cab.

The Sainsmart relays are themselves coils, so they need diodes, too. However, I think that all of the Sainsmart boards already have diodes pre-installed, so it shouldn't be necessary to add your own there.

USB disconnects and other electrical interference

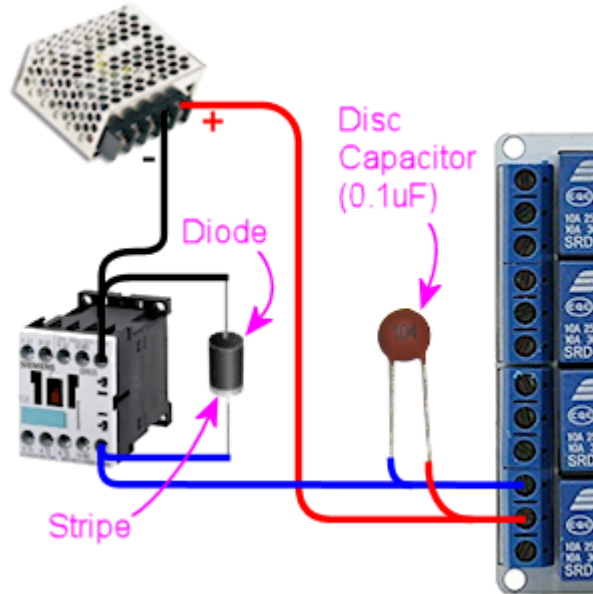
I've seen a lot of reports on the forums from pin cab builders using Sainsmart units and experiencing electrical interference problems.

The most common symptom is probably USB trouble, such as the Sainsmart or some other USB device suddenly disconnecting at random. This can be followed by the device re-connecting by itself after a few seconds, or the device might simply stay disconnected until you manually unplug it and plug it back in. Other symptoms include random keyboard input, random feedback device activation (lights flashing randomly or solenoids firing randomly), or the PC suddenly rebooting.

If you're having any of these symptoms, the first thing to check is that all of your mechanical feedback devices are equipped with diodes. See Chapter 53, Coil Diodes. Devices like solenoids, contactors, and motors cause electrical interference on the power wiring, which can usually be suppressed sufficiently by properly installing a diode on each device's power connection. The Sainsmart relays themselves have the same issue, since they use magnetic coils, but you don't usually have to worry about those because most of the boards come with suitable diodes integrated into the design. Even so, you should check to make sure that your boards do in fact include a diode for each relay, and if not, you'll need to add one.

The second thing to try (after ensuring all necessary diodes are installed) is to add a small capacitor on each Sainsmart relay switch. Mechanical switches create their own kind of electrical noise (called "bounce") during the brief instant when the contacts come together or separate. This can be significant when a relay switch controls a high-current device like a coil, motor, or large LED. You can mitigate this by adding a small capacitor across the switch contacts.

Try a capacitor around $0.1\mu\text{F}$ (which is the same as 100nF). Use a **ceramic disc** type capacitor. These aren't polarized, so it doesn't matter which direction you install it.



If a 100nF ($0.1\mu\text{F}$) capacitor doesn't help, or seems to reduce the problem but doesn't fix it entirely, you might experiment with other nearby capacitor sizes, from 200nF to $1\mu\text{F}$.

Whereas diodes protect against the special type of interference that magnetic coils generate, capacitors help suppress the transient noise caused by switch contacts connecting and disconnecting. So capacitors might be needed for any type of feedback device controlled by a relay switch - even something like an LED that doesn't need a protective diode. However, I wouldn't bother installing any switch capacitors unless you're experiencing a problem that you can isolate to a particular device, and that device is already protected with a diode (or doesn't need one).

If the device in question is a motor, and the problem isn't solved by adding a diode and a capacitor, there's one more thing you can try: an inductor filter, also known as a choke. Capacitors act as voltage filters, leveling out little spikes in voltage changes, whereas inductors act as electric current filters, smoothing out sudden changes in current flow. Motors can sometimes benefit from the latter. See "Electrical interference" in Chapter 61, Shaker motors for parts selection and a wiring diagram.

Quick summary:

- Diode:
 - **Required** for anything with a coil (solenoid, contactor, relay, motor)
 - **Not required** for non-coil devices (LEDs, lamps)
- Capacitor:
 - **Not required** for any device
 - But **might be helpful** for any type of device that causes interference

- Try ceramic disc capacitors with values 100nF to 1μF
- Inductors:
 - **Not required** for any device
 - But **might be helpful** for motors that cause interference
 - Try inductors around 4.7μH
 - Use one or two per motor, wired in series with the power wiring to the motor

Fuses

Some cab builders include a fuse in each SainSmart output circuit. Fuses can help protect an output controller against wiring faults or device malfunctions. I'm usually pretty conservative on these things, so this might be surprising, but I don't think fuses are all that useful with SainSmart outputs. The reason is that these outputs use relays, so they're pretty tough to start with. Fuses are certainly appropriate for solid-state devices like an LedWiz, but I don't think they add much useful protection for a relay board. If want to add them anyway, see Chapter 84, Fuses.

DOF Setup

If you're using my DOF R3++ version, it includes automatic SainSmart board detection and configuration. You shouldn't have to do any of the additional config file setup described below. Just run DOF and it should automatically find your SainSmart board. You can skip straight to DOF Config Tool setup below.

With older DOF versions, SainSmart boards require some extra setup to use with DOF.

If you haven't already set up DOF on your cab, follow the instruction in Chapter 46, DOF Setup.

After DOF is set up, you have to create and edit a file called **Cabinet.xml** in the **DirectOutput > config** folder. See Chapter 46, Extra controller setup in Chapter 46, DOF Setup for the basics on getting this file started.

If you're using the template Cabinet.xml file provided with my DOF setup instructions, you should find a section with a pair of lines like this:

```
<OutputControllers>
</OutputControllers>
```

If you got your Cabinet.xml file from another source, these lines might be missing or might have other contents in between. If they're missing, add them directly under the line(s) that looks like this:

```
<Name>My Pin Cab</Name>
```

There might be different text between the bracketed <Name> markers, but that's okay. What's important is that you add the <OutputControllers> section just below the <Name> section.

If there was already something in the <OutputControllers> section, that's okay, too. You just need to add the new text at the end of the existing text, just **before** the line

that at the end that reads `</OutputControllers>`.

Now copy and paste the following into the file, placing it between the two `<OutputController>` markers:

```
<FT245RBitbangController>
  <Name>Sainsmart 1</Name>
  <SerialNumber>A1234XYZ</SerialNumber>
</FT245RBitbangController>
```

You have to make one more change to that text: you have to edit the Serial Number and replace it with the correct value for your SainSmart board. You might be able to find your serial number printed on your board or the packaging. If not, the board should have come with some software (or a link to download software), which should include a program called FTDIChipList.exe. Run that program. This will list the serial numbers for all of the similar devices in your system.

If the program lists more than serial number, you must have other devices that use the same USB chip set. The SainSmart uses a common chip that's also used in a variety of other devices, so it's quite possible the chip finder program will discover more than one such device in your system. If this happens, and you're not sure which of the devices is the SainSmart board, try the following:

- Exit the chip finder software
- Unplug the SainSmart board
- Run the chip finder software
- Write down the list of chips it reports
- Exit the software
- Plug in your SainSmart board and give it a few seconds to connect
- Run the chip finder software again. Compare the new list to the list you wrote down last time. Since the SainSmart board was unplugged last time, it shouldn't have been included in the previous list, so you should be able to identify the SainSmart board as the new entry that wasn't in the old list.

Once you determine the correct serial number, go back and edit the Cabinet.xml file to replace the fake placeholder text I provided in the template ("A1234XYZ") with the true serial number.

There's one more step. Look for a pair of lines like this:

```
<Toys>
</Toys>
```

As with the previous section, you'll have to add the lines if they're not already there. Add them just after the closing `</OutputControllers>` line. And as before, if this section is already there and already contains other text, simply add the new text at the end of the existing text, just **before** the final `</Toys>` line.

Here's the text we're going to add to the `<Toys>` section:

```
<LedWizEquivalent>
  <Name>SainsmartLWEQ 1</Name>
  <LedWizNumber>40</LedWizNumber>
<Outputs>
```



```

<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.01</OutputName>
  <LedWizEquivalentOutputNumber>1</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.02</OutputName>
  <LedWizEquivalentOutputNumber>2</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.03</OutputName>
  <LedWizEquivalentOutputNumber>3</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.04</OutputName>
  <LedWizEquivalentOutputNumber>4</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.05</OutputName>
  <LedWizEquivalentOutputNumber>5</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.06</OutputName>
  <LedWizEquivalentOutputNumber>6</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.07</OutputName>
  <LedWizEquivalentOutputNumber>7</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
<LedWizEquivalentOutput>
  <OutputName>Sainsmart 1.08</OutputName>
  <LedWizEquivalentOutputNumber>8</LedWizEquivalentOutputNumber>
</LedWizEquivalentOutput>
</Outputs>
</LedWizEquivalent>

```

You can use that exact text, with no changes, as long as you have one SainSmart unit only, and it's the 8-relay type.

If you have two SainSmart units, you first must repeat the <OutputControllers> entry we added above, again substituting the actual serial number for the second device. And second, you must repeat the <Toys> entry we added above, but this time, change the <LedWizNumber> section to **41** (make it 42 for a third unit, 43 for a fourth unit, etc), **and** change all of the **Sainsmart 1.0x** entries to **Sainsmart 2.0x** for the second unit (or 3.0x for the third unit, 4.0x for the fourth unit, etc).

If your SainSmart unit has a different number of relays (they make them with 1, 2, and 4 relays as well), go through the >Toys> text we added and delete all of the sections beyond the number of relays you have. For example, if you have a 4-relay board, you should delete the "1.05" through "1.08" sections.

DOF Config Tool setup

Almost done! The one remaining step is to enter your new SainSmart board into your DOF Config Tool settings. This step is pretty easy compared to the previous one, fortunately.

- Point your browser to the DOF config tool

- Click the My Account tab
- Find "Number of SainSmart Devices" in the list, and set it to "1" (or the number of these boards you've installed, if more than one)
- Click Save Settings
- Click the Port Assignments tab
- In the "Device" drop-down, select "SainSmart 1"
- Go through the port list (Port 1 through Port 8), and select the type of device that you've physically wired to each output port
- Click Save Config
- If you have more than one SainSmart board, repeat the process for each one by selecting the devices one at a time in the "Device" drop-down at the top of the page
- When done, click Save Config, then click Generate Config to download your updated DOF config files
- Unzip the downloaded files into your DOF install folder

All of this is described in the Chapter 46, DOF Setup section as well.

52. LED Resistors

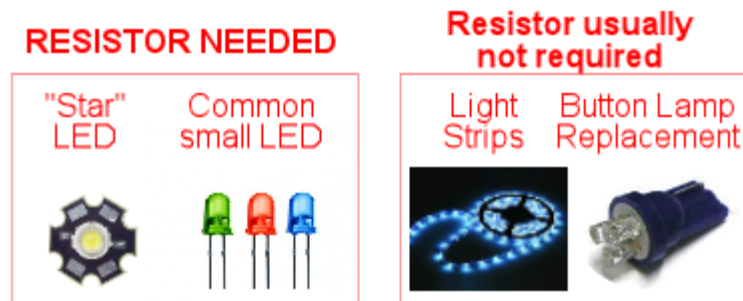
LEDs always require something to limit the amount of current (amperage) going through them. Without some kind of current limiter, an LED acts like a short circuit, which will make it overload itself or its power supply. So it's always necessary to put something in the circuit that sets a safe limit on the LED's power consumption. The simplest and most common way to do this is by placing a resistor in series with the LED.

Which LEDs need resistors?

All LEDs require something to limit current, but some LEDs come with suitable resistors already built in. You don't need to add anything external for devices with their own built-in resistors.

How can you tell if resistors are built-in? A rule of thumb is that any LED that you buy as a bare component requires an external resistor, whereas finished products that happen to contain LEDs usually have any necessary resistors built in.

Here's a handy guide to which is which, for the parts commonly used in pin cabs:



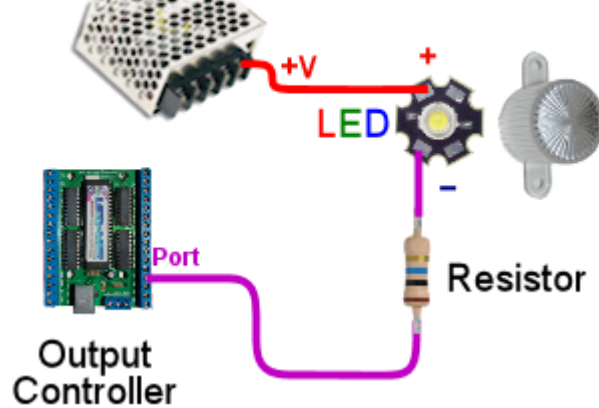
- "Star" LED: **external resistor required**
- Common small LED: **external resistor required**
- Light strips, standard type: resistors are built in
- Light strips, addressable type: resistors are built in
- LED #555 replacement bulbs: resistor is built in
- LED #161 replacement bulbs: resistor is built in

Special exception: You don't need a resistor with any type of LED when using it with a "Small LED" output port on the Pinscape main expansion board. The Small LED outputs are special in that they have their own built-in current limitation, which makes an external resistor unnecessary. This exception doesn't apply to the Pinscape flasher output ports or any other output ports; it only applies to the "Small LED" ports.

How to wire the resistor

Here's the basic wiring plan for connecting a current-limiting resistor to any LED feedback device:

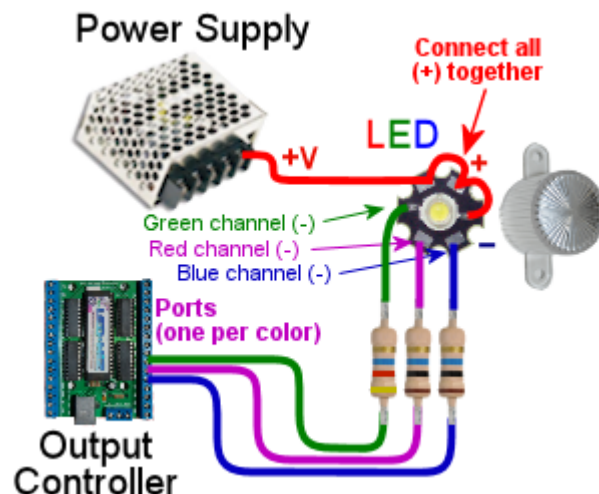
Power Supply



As you can see, this doesn't change the basic wiring plan for an output device, it just adds to it, by inserting the resistor into the wiring between the LED and the output controller.

Note that it's equivalent to insert the resistor into the positive voltage wire (the one that connects the LED to the power supply) instead of the negative wire (to the output controller port). You can do that if you prefer it for some reason. But it's almost always more convenient to put it on the negative side, as shown, because it's usually easier to daisy-chain all of the positive voltage connections together. In fact, for some RGB LEDs, the negative side is the only option because the positive connections for the three color channels are sometimes wired together *inside the LED*. I'd recommend keeping things simple by adopting a uniform rule that the resistor always goes on the negative side.

For an RGB LED, **each color channel needs its own resistor**. In fact, the resistor values for the three channels might even be different. When you're calculating the resistor values (which we'll come to in a moment), you should do a whole separate calculation for each channel. Even though an RGB LED looks like one LED, it actually has three physically separate LEDs inside, each with its own current and voltage specs.



*Basic resistor wiring plan for an RGB LED. This type of LED has separate connections for the three color channels. Each color channel must be wired to its own resistor. **Note:** the order of the pads shown here won't necessarily match your LEDs.*

Choosing a power supply

In most cases, you can use a 5V supply for LEDs.

LEDs should always list a "forward voltage" value, sometimes written V_F . This is usually somewhere between about 2V and 4V. Red LEDs tend to be at the lower end of that range, and blue and green are at the higher end. A common point of confusion for new cab builders is what this voltage means for the power supply. The forward voltage **isn't** the same as the supply voltage: you don't have to use a power supply that exactly matches the LED's forward voltage. Rather, the forward voltage tells you the **minimum** required supply voltage.

The basic rule is that the supply voltage has to be higher than the forward voltage. Ideally, it shouldn't be too much higher, because the higher it is, the more power will get wasted as excess heat burned up by the resistor.

5V is usually the ideal choice, because it's the lowest common supply voltage that's high enough to work with most LEDs.

Choosing the resistor

The exact resistor you need depends on the LED. To figure the right resistance value, you need to know two electrical specs for the LED and one for your power supply:

- The LED's **Forward Voltage**, sometimes written as V_F
- The LED's **Forward Current**, or I_F
- Your power supply voltage, which we'll write as V_{supply}

You should be able to find the LED figures on the LED's packaging, the manufacturer's data sheet, or the eBay seller's page where you bought it. If you buy from eBay, be sure to write down those figures when you place your order, since eBay pages don't always stay around indefinitely.

The power supply voltage is simply the voltage of the power supply you're using to power the LED. It's best to use the **lowest power supply you have that's above the V_F voltage**. For example, if V_F is 3.2V, any power supply higher than 3.2V will work, so you should use your 5V supply.

For an RGB LED, you'll need **three** resistors - one per color channel. What's more, you might need different resistors for each color. In the specs for the resistor, you should find separate V_F and I_F values for the Red, Green, and Blue components. Calculate each resistor value separately, as though it were for a separate LED (which it really is).

Once you have those three figures, you can simply plug them into this formula to calculate the resistance required:

$$R = (V_{\text{supply}} - V_F) / I_F$$

By the way, if you bring up the on-line version of this chapter in your Web browser, you'll find a handy interactive calculator that does this calculation for you, and automatically applies the adjustments described below to round to a standard size. It also calculates the required wattage.

Note that formula takes the Forward Current (I_F) value in Amperes, not milliamps. The LED will probably specify I_F in milliamps. To convert from mA to Amps, simply divide by 1000. For example, if I_F is quoted as 20mA, use 0.02 Amps in the formula; if I_F is 350mA, use 0.350 Amps.

The result R is in **Ohms (Ω)**. It's **not** in $K\Omega$ or any other multiplied value - it's in

plain old Ohms. For a high-power "Star" RGB LED, you might be surprised to get a value in the single-digit Ohms range, like 5.2Ω. That's perfectly normal, so don't second-guess the formula and think the result needs to be in a different unit. As long as you entered the inputs correctly in **Volts** and **Amps** (not milliamps), the result is always in Ohms.

The mathematical result of the formula tells you the exact resistance you'd need in theory to get the desired current. But you can't buy resistors in just any size; they only make them in certain standard sizes. The standard resistor sizes are 1Ω, 1.2Ω, 1.5Ω, 1.8Ω, 2.2Ω, 2.7Ω, 3.3Ω, 3.9Ω, 4.7Ω, 5.6Ω, 6.8Ω, and 8.2Ω, plus each of these values multiplied by 10, 100, 1000, 10,000, 100,000, and 1,000,000.

So after you figure the theoretical R value with the formula, round **up** to the nearest standard size. For example, if the formula says you need a 4.9Ω resistor, round up to the next standard size of 5.6Ω.

We're going to call this rounded-up value the **adjusted** resistance. (Sorry if this is starting to feel like filling out a tax form!)

After you figure the resistance value and round up to a standard size, the next step is to calculate the required wattage for the resistor. This is important because resistors generate heat, so you need to buy a resistor that's capable of handling the amount of heat it's going to generate.

To calculate the wattage, plug the **adjusted** R value (the value rounded up to a standard size) into this formula:

$$I_{\text{actual}} = (V_S - V_F) / R_{\text{adjusted}}$$

That will tell you the actual current you're going to get with the adjusted resistance value. Now plug the result of *that* formula into this formula:

$$W = (V_S - V_F) * I_{\text{actual}} / 0.6$$

That yields the amount of power the resistor uses in Watts, and adds a safety margin so that we never operate above 60% (that's the 0.6) of the maximum rating for the resistor. That's just to make sure we're not pushing our luck, and accounts for any manufacturing variations in the LED or resistor that make them a little outside of their rated specs.

As with the Ohms value, they only make resistors with certain wattage ratings, so you'll have to choose the next available larger wattage. Common wattage values for resistors are 1/8W, 1/4W, 1/2W, 1W, 2W, 5W, and 10W.

Now you're set to go out and buy resistors! Go to an electronics vendor like Mouser and enter the Ohms and Watts values you just calculated. You should be able to find matching parts. If you have any trouble finding something with the required Watts value, it's always safe to use a larger rated wattage. For example, if the formula says you need 5.6Ω at 1W, it's fine to use a 2W resistor instead. Larger wattage resistors are physically larger as well, so it'll take up more space if you have to use a much higher wattage than the formula gives you, but it will be perfectly functional and safe.

53. Coil Diodes

For every mechanical feedback device that you install in your cabinet, you should install a protective diode. This applies to **anything with a magnetic coil**:

- Contactors
- Solenoids
- Replay knockers
- Chimes
- Bells
- Relays
- Motors of any kind (shaker, gear, etc)
- Rotating beacons

It's important to install diodes on these devices, because all of them have magnetic coils that generate voltage spikes as part of their normal operation. These spikes, known as "flyback" current, can be quite powerful and can damage your computer motherboard and other electronics, such as your output controllers and other USB devices. Diodes suppress these spikes and block them from propagating through your system.

Note that these voltage spikes from the coils are unrelated to power surges in your house wiring, the kind that you buy surge-suppressor power strips to protect against. The coil voltage spikes come from the devices *inside your cabinet*, so a surge suppressor on your power strip will provide no protection against them. You always need a diode on each individual coil, to cut off the spike at the source.

Fortunately, diodes are simple to install, and you don't need to know anything about electronics to set them up.

You don't need protective diodes for LEDs, incandescent bulbs, LED strobe lights, or LED light strips.

Diodes are also used in the Pinscape expansion boards, and have many other uses besides coil protection. This chapter is only concerned with the coil protection usage. If you want to know more about diodes as circuit board components, see Chapter 74, Diodes.

Don't use diodes with AC devices

Everything we say here applies only to **DC devices**, using DC voltage as the power supply. **Don't** use diodes this way with devices that run on AC power (for example, anything that plugs directly into a wall outlet for 120V or 240V mains power). Doing so will cause a short circuit that will blow a fuse (or worse).

Choosing a diode

I'll just cut to the chase here: 1N4007.

1N4007 diodes are cheap (15¢ apiece) and have pretty ideal electrical specs for this job. Buy one for each coil-type device in your cab.

Like any other engineering question, there are different schools of thought on this, and you'll probably be able to find competing recommendations if you ask around.

But the 1N4007 is an excellent general-purpose choice that works well for everything in a pin cab. If you look at the real pinball machines made in the 1980s and 1990s, they all use 1N400x diodes for this same job.

(One question that's come up in the forums is whether 1N4007 diodes are suitable for high-current devices, such as 4A knocker coils or 3A shaker motors, when the 1N4007 data sheet says the "forward current" limit is only 1A. The answer is yes, 1N4007 is still suitable for this! As we'll see below, we're not actually using the "forward current" capacity of the diode to carry the 4A knocker coil current or the 3A shaker motor current, because we're going to install these diodes *backwards*. The diode will only carry the "flyback" current from the coil, which is quite different from the supply current when the coil is operating. The 1N4007 has good properties for this job because it has a very high reverse voltage rating [700V], which makes it safe to install backwards for high-voltage devices, and it has a high rating for momentary surge current [30A], which is precisely the nature of the flyback current.)

How to install them

Each coil-type device in your system should have its own diode. That's one diode per motor, one per solenoid, one per contactor, and so on.

Type of device: Use a diode for everything with a coil, magnet, motor, or really anything with moving parts: relays, contactors, solenoids, knockers, motors, bells, chimes, rotating beacons. You **don't** need to add diodes to lights with no moving parts (incandescent lamps, LEDs, LED strips, strobes, flashers).

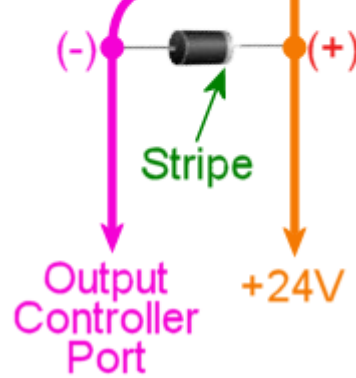
Location: Install each diode as close to the coil as is convenient. The diode's purpose is to shunt the voltage surge from the coil back into the coil, to keep the surge from propagating into the rest of the wiring. As such, you want it physically close to the coil. I recommend installing the diode directly across the coil's terminals or connecting wires.

Don't worry about getting *extremely* close to the coil. I've talked to a few people who took the advice about getting close to the coil very seriously, to the extent of disassembling the packaging around a motor or solenoid so that they could get a few centimeters closer. It's really not necessary to do that. Just attach the diode at a convenient place where the device's terminals or wires are exposed for connecting externally. The same terminals where you attach the power wires to the coil are perfect for the diode.

Orientation: The striped end of the diode **must** be attached to the positive (+) supply voltage to the device. The other end must be attached to the (-) terminal, which you'll connect to a port on your output controller.

Orientation is critical! Always double-check that you get it right. If you get it wrong, it'll create a short circuit as soon as you turn on the device, which will most likely damage your output controller.





"But doesn't the stripe on a diode always go to minus???" In *normal* uses for diodes, you're exactly right. But this is a somewhat unusual use case, and in this case we really do want the diode to go **against** the normal flow of current. The striped end really must go to the **positive** voltage.

How to attach them: The best way to attach a diode to a particular device can vary depending on the type of terminals it has. No matter what, though, I recommend always **soldering** the diode to *something*. This creates a permanent connection, which is much more reliable than temporary connections like screw terminals. It will also help protect against accidentally reversing the diode if you should ever have to disconnect the wires, since the diode itself will be permanently installed.

If you're using contactors, or anything else with screw terminals, I recommend soldering the diode to your wiring. Put it a few inches from the ends that you connect the screw terminals.

For devices with solder terminals, such as pinball coils and DC motors, you can usually solder the diode directly to the device's terminals. If they're not close enough together, simply add a short piece of wire to connect one end.

If you're using a genuine pinball coil, such as a replay knocker, it might already have a factory-installed diode. If so, you don't need to add another one.

Wherever the diode ends up, you should take care that the bare wire leads of the diode can't come into contact with any other metal or wires. I recommend wrapping any exposed wire or solder with electrical tape to insulate it.

My system works without 'em! Why bother?

Some people build their systems without ever hearing about the need for diodes, and sometimes they get away with it, with no obvious problems appearing. But even if your system seems to work without any diodes installed, you're running the risk of something breaking in the future. You're putting stress on your other components every time a coil or motor switches off.

The coil surge current isn't always big enough to fry everything instantly. You can often get away without diodes for a while without any obvious problems. But the surge current is happening nonetheless, and chances are that it's causing incremental damage that will eventually make something break.

In some cases, missing diodes can cause symptoms that appear to be software problems. The most common issues that can often be traced to missing diodes are phantom keyboard input (keys seemingly pressed at random), and intermittent USB device disconnections. If you're seeing anything like that, make sure your coils all have diodes installed, and that the diodes are installed correctly.

Do I really need this with a motor?

There's a common notion that the surge current from a motor comes from the "generator effect" of the motor. As you probably know, any electric motor can also serve as a generator: turn the shaft with an outside force and you'll generate some electricity with the motor. So you might reasonably think that the motor's momentum will cause it to keep spinning for a few moments after you turn off the power, making it generate some residual electricity. Is this what causes the voltage spike in a motor?

The generator effect is real, but no, it's not the source of the spike that we're worried about. The generator effect voltage is too low to be a problem.

Motors have magnetic coils - that's what makes them go - and these coils have the same physics as solenoid coils. That means they have the same surge current as other coils. This is completely separate from the mechanical action of the motor; it's a purely electromagnetic effect, and it causes the same problems in motors that it does in other inductors.

The main reason I point this out is that it's easy to talk yourself out of adding a diode to a motor if you think in terms of the generator effect alone. You might look at the motor and decide that it just doesn't have enough momentum for this to be a problem. But that misses the more important point that you need a diode for a motor anyway, simply because it's an inductive device with a magnetic field.

Theory of operation

If you're interested in learning more about the physics behind this, read on. You can skip the rest of this section if you only care about the practical dimensions. Just install the diodes as outlined above and you'll be set.

How it works

If you know a thing or two about electronics, you might have noticed that the diode is installed "backwards" from how you'd normally use it, in that we have the stripe attached to the positive side.

Good catch if you noticed that, but it's not an error! We really do want the diode to be installed **opposite to the normal current flow**. Why? Think about what would happen if it were installed the other way: when the power goes on, the diode would happily conduct all of the current straight through, bypassing the load. In other words, it would create a short circuit from the power supply directly to the output controller. This would instantly fry something - the diode, the power supply, or the output controller - with the unrestrained current.

With the diode installed opposite to the flow, though, it doesn't conduct at all when the power goes on. It's like it's not even there. All of the power goes through the load (the feedback device) just like we want it to, and nothing gets fried.

So if the diode never conducts, what good is it? Well, it's not quite true that the diode *never* conducts. It never conducts *in the power supply direction*. But it does kick in when the power turns **off**. That's when the coil releases the current surge we've been talking about. Due to the physics of magnetic fields, it turns out that the surge current goes in the opposite direction of the original current that created the field in the first place. Basically, the energy that gets stored in the magnetic field by the power supply current comes rushing back out in the opposite direction when you take the power away, like the air coming back out of a balloon if you stop inflating it. The surge current is going "backwards", and the diode is installed "backwards", so

the surge current is actually forwards from the diode's perspective. The diode thus allows the surge current through, sending it back into the coil. This blocks it from flowing down the other wires that go to the power supply and the output controller.

The electrical resistance of the coil wiring quickly turns the surge current into heat, safely disposing of it. So the surge fizzles out without damaging any sensitive components at the ends of the wires.

There might appear to be a couple of contradictions in what we've just said. Let's address any lingering doubts. First, if it would have fried something to run the *original* current through the diode, why doesn't it fry something when we run the *reverse* current through it? The answer is that the total energy in the reverse current is much lower, because it's not being driven by a power supply; it's limited to the energy stored in the magnetic field, which is fairly small in absolute terms. The surge does heat up the coil a tiny bit - that's where the excess energy goes - but only a tiny bit. Not enough that you'd be able to feel it by touch, and not enough to do any damage. Second, if the flyback current is so dangerous, why doesn't it hurt the coil, or the diode itself? In this case, the answer is in the different natures of the different components. Transistors and IC chips are extremely sensitive to voltage, even at very low total energies, because their internal structures are so tiny. Exposing them to high voltages can punch holes in their internal structures and destroy them. Coils and wires, on the other hand, are relatively indifferent to voltage levels as long as the total power is limited. The point of the diode is to isolate the surge current so that it stays inside the coil - and away from your other circuitry - since the coil isn't affected by momentary high voltages. As for the diode itself, it's a perfect gate-keeper, because the 1N4007 can handle high voltages *and* high momentary current surges, which is exactly what the flyback current looks like.

Where the surge current comes from

When you send electricity through a coil of wire, the moving electric charge induces a magnetic field in the region around the coil. For devices like solenoids and motors, the magnetic field is the whole point, because it's what converts the electrical energy into mechanical action.

The inductive effect converts the energy going into the coil from electric to magnetic energy. Most of that magnetic energy goes straight into the mechanical action that the device is designed to produce, such as spinning the motor or moving the solenoid plunger. But a portion of the energy goes into the field itself. So the field contains a certain amount of energy as long as it's standing.

When you switch off the electricity, you stop feeding energy into the magnetic field, so the field can no longer sustain itself and starts collapsing. At this point, the energy contained in the field has to go somewhere. The most direct path for the field energy to escape is straight back into the coil wiring. Like many processes in physics, induction works in both directions: an electric current induces a magnetic field, and a magnetic field induces an electric current. The collapsing magnetic field induces a current through the coil wiring. This "field collapse" current moves in the opposite direction of the original current (from the power supply) that created the field.

The current induced by the magnetic field collapse is the surge we've been talking about. The thing that makes it harmful is that the field collapse happens very quickly. When the electricity shuts off, the magnetic field has a Wile E. Coyote moment where it suddenly realizes it's suspended in mid-air, and instantly plummets to the ground. In this case, the magnetic field energy escapes rapidly into the coil, transferring all of its energy to the coil in a few milliseconds. The sudden



surge of charge drives the voltage very high. For a 12 Volt coil, the surge can spike to 300 or 400 Volts.

Even though that's a very high voltage, it's not typically a threat to human safety, because the total amount of energy in the collapsing field is relatively small. The voltage gets so high only because the surge is so brief. But a brief high voltage *is* a threat to certain types of electronic components, especially the microelectronics in integrated circuit chips. Those devices are physically so tiny that it doesn't take much energy to damage them. It only takes a brief blast of high voltage. It's the same thing that makes static electricity discharge such a danger for many electronic devices.

You can find more about this in the Wikipedia article about flyback diodes.

54. Coil Timers

If you're using any real pinball coils in your virtual pin cab, you might want to consider using some kind of "time limiter" circuit with them. The reason is that most types of pinball coils are only designed to be energized for very brief intervals, typically just a fraction of a second at a time, and can overheat if energized for too long at a time. Overheating will melt the wire inside the coil and destroy the device.

Examples of pinball coils where this is an issue:

- Replay knocker
- Bumpers
- Slingshots
- Chime units

This is actually a fairly common property of solenoids in general, not just pinball coils, so it might apply to your pin cab if you're using other types of devices to simulate any of the above:

- "Open frame" solenoids bought on eBay or elsewhere
- Automotive starter solenoids

How a time limiter circuit works

A time limiter circuit can be used as a fail-safe to protect your coil devices against getting stuck on for long periods. A time limiter adds some special additional circuitry to an output controller port that automatically limits the amount of time the port can be "on" continuously. If the port stays on for longer than the limit, the timer circuit forces the port to turn off, overriding any software signal to the contrary.

The reason to implement this with a hardware timer, rather than with some kind of software fix, is that the main thing we're worried about is software failure. A separate hardware circuit won't be affected by any errors or crashes in the software.

The Pinscape expansion boards include built-in time limiters for the Replay Knocker port and for all ports on the Chime boards. You can add your own custom timer to any other type of output controller as well, which we'll describe below.

Why this is a concern

There are two places in a virtual cab where coil overheating can be a problem: flippers, and everywhere else.

For flippers, it should be fairly obvious why this is an issue. The player can keep a flipper button pressed indefinitely to trap a ball, consider a shot, have a beer... The flipper coil will stay energized this whole time. It might seem like a typical player wouldn't typically hold the button on all *that* long, but it's not the typical case we have to be concerned about. We have to consider the longest time that anyone will ever leave hold the button on, since all it takes is that one time to fry the coil.

For bumpers, slingshots, replay knockers, chimes, and everything else, there's no reason in *normal* operation that any of these will ever stay on for more than a brief instant at a time. So it might seem like you don't have to worry about them. But you do, because of the chance of abnormal operation. Specifically, the devices in a real cab are controlled by software, and software can fail unexpectedly. There's a

particular failure mode in Visual Pinball where VP can crash just after turning on a device, leaving the device stuck on until the user can intervene manually. This isn't just a random fluke, either; if it were, it wouldn't be likely enough to worry about. The reason it's a concern is that it's relatively likely to happen in this particular way (device turns on, then VP crashes) because of the way VP scripts are constructed.

Devices that you *don't* have to worry about

Not all coils or solenoids have this problem:

- Contactors. These are large electrical relays designed for high-power switching applications. They're specifically designed to be activated continuously for indefinite periods (even for months or years) without overheating. They're popular for simulating flippers in virtual cabs for exactly this reason.
- **Older** pinball flipper coils, for machines made before about 1990. On the real machines, they had to deal with exactly the same issue that we do when it comes to the flippers: the player can press the button and hold the flipper on for long periods. So the pinball manufacturers came up with various ways to prevent the coils from overheating. The technique they used up until about 1990 also works well on virtual pin cabs, so flipper assemblies for those older machines are usually safe to use on a pin cab without any special timer protection. See "Flipper assemblies" below for more on the different types.
- Some other solenoids. Some non-pinball solenoids are built for long duty cycles and others aren't. If you have a data sheet for your device, it might list the maximum continuous "on" time or maximum duty cycle. If not, you can test it. Apply power and carefully monitor the device's temperature. You don't need a thermometer; it's enough to check if it's getting hot to the touch. If it starts getting hot, cut power immediately. If it got hot in a few seconds, you definitely want to do something to limit the device's "on" time. If you can leave the device continuously on for a few minutes and it's not getting noticeably hot, it should be safe to use without any time limiter. If it's somewhere in between, you should err on the side of caution and limit its "on" time.

Flipper assemblies

As mentioned above, *some* flipper assemblies are safe for use in virtual pin cabs, without any special timer protection. Real pinball machines have to deal with the same problem we do when it comes to flippers - that the player might press and hold the button for long periods. The manufacturers came up with a variety of solutions over the years.

The standard approach for many years was to use **two** coils for each flipper: a high-power "lift" coil and a low-power "hold" coil. The lift coil fires briefly when you first press the button, to carry out the the rapid "flip" action. As soon as the flipper is all the way up, the lift coil cuts out, and the low-power hold coil takes over, to keep the flipper flipped while you hold the button. The hold coil had much less lifting power - just enough to hold the flipper up. Less power means less heat, so the hold coil stays cool enough that you can leave it activated indefinitely. Problem solved.

Note that the two coils were always wound together into one physical unit, so it *looks* like there's only one coil involved. The giveaway is that the coil has three terminals instead of the usual two.

Up until the early 1990s, the two-coil flipper assemblies switched between the "lift" and "hold" coils using an end-of-stroke switch embedded in the flipper assembly. The coil was wired directly to this switch. When the flipper flipped all the way up, it hit the switch, cutting off the lift coil power and activating the hold coil power. This system works well in virtual pin cabs, since the flipper assembly itself takes care of

switching between the two coils.

Starting in the 1990s, Williams changed to their "Fliptronic" system. This also uses the two-coil design, but the end-of-stroke switch is no longer wired directly to the coils. Instead, it's wired to the CPU, and the CPU decides when to switch between the two coils. The Fliptronic assemblies **aren't** safe with a virtual machine, because the coil switching is no longer part of the flipper assembly - it has to be directed by the software. The pinball emulators on the PC don't have the necessary programming. If you connected one of these newer Fliptronic assemblies to DOF, DOF would just leave the "lift" coil on the whole time, and we'd be back to the overheating problem.

Newer Stern flipper assemblies also aren't safe for virtual pin cabs, because they use an even newer design that uses just one physical coil and uses software to reduce the coil power during the "hold" period.

To see if you have a "safe" flipper assembly using the two-coil design, first check to make sure there's an "end-of-stroke" switch. If there's not, you probably have a newer Stern assembly that's not safe for virtual use. If you *do* have an end-of-stroke switch, inspect its wiring. Check that it's wired directly to the flipper coil. If it's only wired to an external connector, you have a newer Fliptronic assembly that's not safe for virtual use. If it's wired directly to the coil, you have the older type that *is* safe for a pin cab.

If you have one of the newer types, don't panic! There's still a way you might be able to make it work. If you're using the Pinscape boards, you can enable the Pinscape "Flipper Logic" feature, which uses the same PWM power reduction approach that the newer Stern machines use. Flipper Logic is handled directly in the Pinscape firmware, so it doesn't require anything on the PC to be aware of it, meaning it's compatible with all PC software that can access the flippers at all. See "Flipper Logic" below for more details.

Pinscape expansion boards

The replay knocker output on the main Pinscape board has a time limiter circuit built-in. All outputs on the Chime boards feature the same time limiter. (The time limiter is the whole point of the Chime boards.)

Connect devices to the timer-protected output ports exactly the same way you would connect devices to any of the other Pinscape output ports. The timer function is built into the port's hardware and is always active. There's nothing to configure and no software setup required.

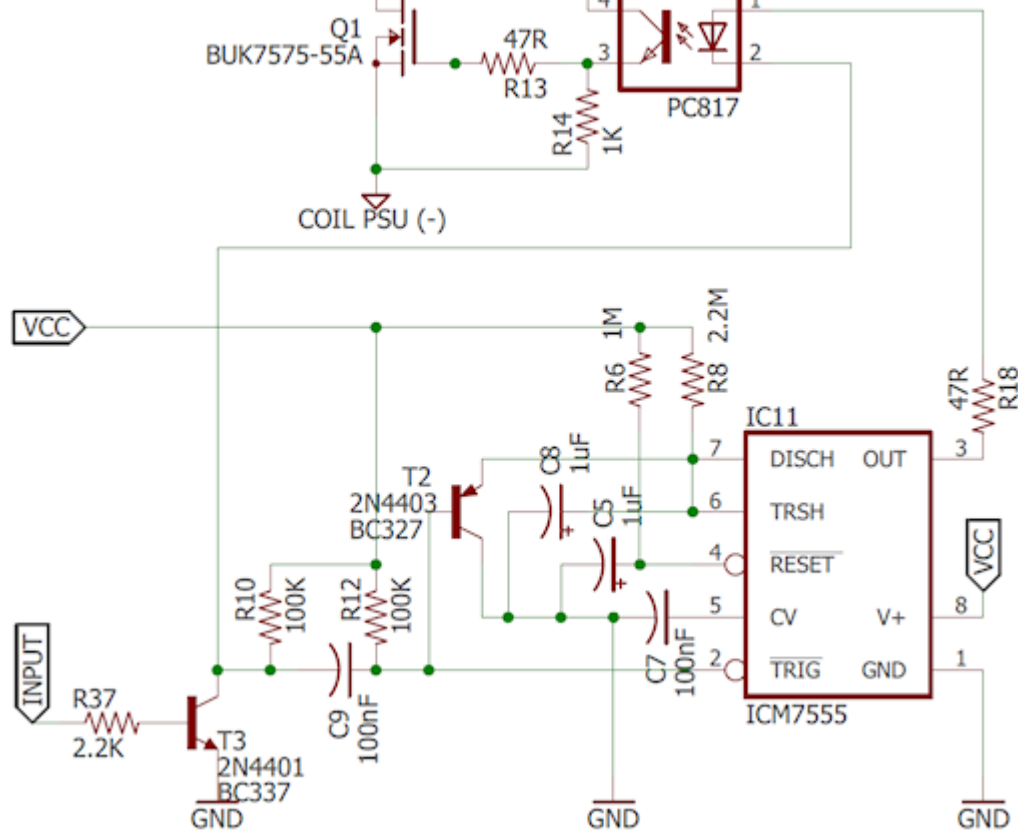
The timers on these output ports cut off power after about 2 seconds of continuous "on" time. They automatically reset as soon as the port turns off.

The short time limit is designed for devices like replay knockers, chime units, bumper coils, and slingshot coils. In normal use, these devices always fire momentarily. The instant reset allows the devices to be fired in quick succession, such as when the ball is bouncing rapidly back and forth between two bumpers.

DIY time limiter circuit

Here's the schematic for the coil timer circuit used in the Pinscape expansion boards. This is a standalone circuit that doesn't depend on anything else on the expansion boards; it can be used with any type of microcontroller to control just about any kind of coil.





External connections:

- **VCC** is the positive voltage supply for the microcontroller. This circuit will work equally well with 3.3V and 5V microcontrollers.
- **GND** is the "ground" for the microcontroller power supply. Connect this to any GND pin on the microcontroller.
- **Input** is the GPIO pin on the microcontroller that will be controlling the device. This is a **high** trigger circuit, meaning that the GPIO line must be driven high (to VCC voltage) to turn the coil on.
- **COIL-** is the negative terminal of the coil to be controller. Connect the other terminal of the coil directly to the coil power supply's positive voltage.
- **COIL PSU (-)** is the coil power supply's negative or GND terminal.

This circuit is designed to be used with a microcontroller, so the input takes a "high" voltage value (the same voltage as VCC) to turn the coil on.

The circuit as shown includes a high-current MOSFET that controls the coil, so you can connect this circuit directly to the coil without any external relays or amplifiers needed. The MOSFET shown in the schematic will handle 14A at 55V, which is enough for any pinball coil.

Adapting for LedWiz use: The circuit could be adapted for use with an LedWiz or similar device, but I'm going to leave this as an exercise to the reader, because I haven't actually built or tested an LedWiz interface version. There are two changes you'd have to make:

- First, you must change the input to trigger on a "low" voltage value, since that's how the LedWiz signal works. You can do this by adding an optocoupler to the input stage. Connect the optocoupler LED anode (+) to +5V, connect the cathode to a 100Ω resistor, and connect the other end of the resistor to the LedWiz output. Connect the Input line in the schematic above to the optocoupler emitter, and also connect it to ground through a 1K resistor.

Connect the optocoupler collector to +5V. Use +5V for VCC throughout.

- Second, you'll have to "smooth out" the PWM signal produced by the LedWiz. The circuit is designed for a simple digital logic high/low GPIO output from a microcontroller. The LedWiz produces a pulsed signal that's never 100% on. Every pulse will reset the timer circuit, which will prevent the timer from ever expiring, which will defeat the entire purpose of the timer. To overcome this, I think you could use a simple low-pass filter on the input from the LedWiz. I think a simple first-order filter, with one resistor and capacitor, would do the trick. You'd have to choose the filter parameters according to the actual controller device you're using; the LedWiz PWM runs at about 50 Hz.

If you successfully build and test an LedWiz-compatible version of the circuit, I'd really like to hear about it! Send me the actual circuit that you got working and I'll publish it here instead of this "exercise for the reader" dodge.

Pre-built timer device

You can buy a pre-built timer device on eBay that provides similar functionality, and put it in your output circuit between your output controller and the coil you want to protect.

The product offerings and sellers on eBay are constantly changing, so I can't provide you with a link to a specific product; if I did, it would just be a dead link by the time you read this, and that wouldn't be very helpful. So I'll have to tell you how to search for it yourself. I'll warn you, though: this will take some persistence on your part, and maybe some trial-and-error buying, because the relevant eBay listings can be difficult to decipher. The products you're looking for will all be from Chinese sellers who don't always write perfect English ad copy on their pages.

The basic search term that usually works best is:

multi-function relay timer board

In very vague and general terms, you're looking for:

- a relay to control an output
- with adjustable timers
- and a big list of different timing modes

These will usually have 15 to 20 different "modes" or "functions" listed. They'll say things like "Function 1: timing on: after power, time delay pull relay T1, between adjustable 0.1 seconds 270 hours, CH1 interface to high level signal, repeat function", and then a giant list of similar tortured sentence fragments.

That much is pretty easy to find on eBay. What's a little harder is making sure that the list of 15-20 modes includes the exact mode we're looking for. In my experience, anything that has a big list of modes like that *will* have the right mode in there somewhere, so if you're feeling lucky and don't want to make your brain hurt parsing Google Translate output, just buy one and give it a try. But if you don't want to risk the \$20, you'll need to scan through the modes and verify that there's one that does what we need.

So I'll try to describe the specific mode you need, in terms that hopefully *are* comprehensible. **Don't** look for my exact words. I'm trying to use sensible English, which the ads generally don't. I'm trying to give you an understanding of what the mode is actually supposed to do, on the theory that if you can internalize this with a solid mental model of the desired function, you'll have a better chance of fitting the poor descriptions on eBay to your good internal understanding. So here's what

you're trying to accomplish with the device:

- You fire a trigger signal (by "you" I really mean "the LedWiz" or whatever: the point is that the signal is being fed into the relay board from an external source)
- In response to the trigger signal, the relay board immediately turns the relay output on
- After a timer expires, the relay board turns the output off, *even if the trigger signal is still on at that point*
- When the trigger signal turns off, the cycle resets, so that the board is now ready to receive a new trigger signal and start the whole process over

As an example, here's some actual verbiage I just pulled from a random eBay ad for a random device that does what we're looking for:

Function 14: Disconnect and then pull the trigger timing: After power relay does not act, a high-level interface to CH1 pulse signal, immediately pull the relay, the relay off delay time T1 after arrival; T2 arrive after disconnecting time relay, delay time between T1 and T2 in 0.1 seconds -270 hours adjustable, repeat CH1 interface to a high level pulse signal, repeat the above functions;

Again, *don't try to look for these exact words*. This is just one example. Every eBay seller will use their own words for this, and they'll all be different, and they'll all be bizarre and hard to read in their own way. But let's pull this particular example apart, as a practice exercise and to help illustrate how this relates to the functions we're looking for:

- "After power relay does not act": this is telling you that the timer doesn't do anything special when first powered on. Weird that they say what it *doesn't* do, right? Well, they're just spelling this out because a lot of the other functions of these boards *are* all about doing something special when the power first comes on. So they're trying to be helpful by saying explicitly that this function isn't related to the power initially coming on.
- "...a high-level interface to CH1 pulse signal": this is the user-generated trigger signal. "CH1" probably stands for "channel 1", which is the input that you'd connect to the external signal source, such as the LedWiz output we want to use as the control. Some of these boards have more than one trigger input, so they might have "CH1" and "CH2" for the two inputs. We only need one input for our purposes, but if the board has two or more, that shouldn't be a problem, as long as there's a mode that operates on the basis of a single input. This particular mode's description doesn't say anything about any other inputs being involved. "High-level" means that the board's input terminal senses the trigger when the connected voltage goes "high", meaning it goes from 0V to 5V or whatever the supply voltage is.
- "...immediately pull the relay": the relay turns on. Just what we want. I'm not sure why the verb is "pull"; maybe they're thinking of the relay's magnetic coil turning on and tugging on the switch contacts to toggle the switch. I bet it makes more sense in Chinese. The Google Translate round trip probably looks hilarious when you translate it back.
- "...the relay off delay time T1 after arrival": the relay turns back off after a delay. "T1" is one of the programmable delay times.
- "T2 arrive after disconnecting time relay...repeat": everything resets after the further delay time T2, also programmable. In other words, you get a forced "off" time of T2 before you can fire the relay again. That's not part of what I

said to look for, but we can effectively ignore this by setting T2 to a very short delay time (minimum 0.1 seconds according to this ad).

High-level inputs: As with my "DIY timer circuit" above, most of these relay boards require a "high-level" input, meaning that they trigger when the input signal changes from 0V to 5V (or whatever the logic voltage is for the board). And as I mentioned with my DIY timer circuit, that's the opposite of how LedWiz's work. See "Adapting for LedWiz use" in the DIY timer circuit section above for some pointers on interfacing to an LedWiz.

Slow-blow fuses

"Slow-blow" fuses are designed to perform a similar function to the type of timer we've been talking about. Rather than blowing immediately on overload, a slow-blow fuse is designed to tolerate an overload for a certain amount of time, eventually cutting off power if the overload goes on too long.

I'd consider a slow-blow fuse to be a good back-up to a timer. Add one if you want to be extra-cautious, just in case the timer doesn't do its job. But I wouldn't consider a fuse to be a good replacement for a timer. The problem with fuses is that they're not precision devices; they don't blow after precise time periods or when exact loads are exceeded. You can count on a timer to have precise timing characteristics; you can't expect the same thing from a fuse. And, of course, fuses are expendable, and must be replaced if they ever actually do stop an overload; a timer can be triggered over and over without any harm done.

See Chapter 84, Fuses for information on selecting slow-blow fuses and how to predict their timing and loading properties. That chapter includes a section on pinball coils and how to select appropriate slow-blow fuses for them.

Pinscape "Flipper Logic" setting

The Pinscape Controller firmware has a Flipper Logic feature that can be used to provide similar timer protection at the controller firmware level.

The Flipper Logic feature is implemented in software, so I wouldn't consider it as bulletproof as a dedicated hardware timer. Software is just categorically less reliable than hardware. But I'd consider it far more reliable than software on the PC, since it runs in the KL25Z, which is a much more isolated and controlled environment. So it's a good first line of defense, and much better than nothing. If you're using the Pinscape software, and you don't want to go to the extra trouble and expense of implementing hardware timers, you can greatly reduce your coils' exposure to overload by enabling the Flipper Logic feature for them.

What Flipper Logic does: The feature wasn't actually created as an alternative to coil timers, but it can serve that purpose anyway.

What it was actually created to do was to emulate the "lift" and "hold" power settings in real pinball machine flipper assemblies. As we mentioned above, most real pinball flippers generally use two physical coils, a high-power "lift" coil, and a low-power "hold" coil, which are engaged, respectively, as the flipper is first activated and then held.

Some newer Stern machines do the same thing with a single physical coil, simulating the low-power "hold" coil by reducing power to the main coil. That's exactly what Flipper Logic does. Flipper Logic provides full power to a port for a given initial interval immediately after the port is first activated, and then reduces power to a lower PWM level if the port is kept on beyond that initial period.

How to use it as a coil timer: The trick to using this as a software coil protection timer is that the "hold" power can be set to zero, cutting off power to the coil entirely. When the hold power is set to zero, it's effectively the same thing as a coil timer. The only difference is that it's implemented in software rather than as a dedicated hardware circuit.

How to use for flippers: For flipper emulators, you don't want the flippers to un-flip when the time limit expires. You want them to act like real flippers and stay flipped, just with reduced "hold" power, to avoid overheating. So you want to find lowest hold power level that keeps the solenoid activated. The required power level is a function of the specific coils or solenoids you're using, so you'll have to experiment. Start at the lowest setting, and turn it up until the flipper stays flipped. Once you find that level, test it to make sure it doesn't overheat the coil! Carefully monitor the coil while holding the flipper on - you can just check to see if it's getting hot to the touch. If it stays cool after being on for a couple of minutes, it should be safe indefinitely. Note that a little heating is okay, but it shouldn't ever start feeling hot.

Enabling Flipper Logic: Flipper Logic can be enabled individually on each output port via the Pinscape Config Tool. Go to the Settings page and scroll down to the Output Ports section. In each port, you'll find a little "flipper" icon in the group of control icons at the right side of the page. Click the flipper icon. This will let you enter two parameters: the initial full-power time allotment, and the "hold power" PWM level. To use this as a coil timer, set the initial full-power time to a suitable value (around 500ms should be good for any momentary device like a bumper, slingshot, knocker, etc), and set the hold power level to 0.

Remember to click "Program KL25Z" after you've finished making changes, to save the new settings to the controller board.

55. Button Lamps

Most of the buttons on a pin cab can be illuminated. Any button with a lamp can have the lamp controlled by software, too, so that the button lights up, turns off, blinks, and fades in and out when appropriate.

Examples of illuminated buttons on a pin cab:

- The Start button on a newer (1990s+) machine usually lights up when a new game can be started, and sometimes blinks
- The Launch Ball button can light up or blink when a ball is ready to launch
- Some machines have an Extra Ball button that lights up when the "buy-in" feature is available
- The flipper and Magna Save buttons can be illuminated and can be set up to show different colors for different games, to match the original flipper button colors of the arcade versions
- The coin slots in most coin doors are illuminated

Button lights are optional, but most cab builders include them because they make the machine seem more alive. And controlling the buttons through software is also optional (you can just hard-wire them to be constantly on instead), but most cab builders want them actively controlled, since that makes the machine more interactive and more realistic.

If you're not sure what all of the buttons are for, see Chapter 34, Cabinet Buttons for more detailed descriptions of the various button types commonly found on virtual cabs.

Hard-wired vs software controlled

Some pin cab builders simply wire all of the button lamps so that they're always on. That's the easy way to do it, certainly, but most cab builders want the software to be able to control each of the buttons, so that they only light up when appropriate. To do this, you need to think of the button lamps as output devices, just like shaker motors and solenoids.

If you wire the button lamps to your output controller, DOF can control them in sync with the game action, just like other output devices. All of the common buttons are specifically supported in the software, so all you have to do is set up your DOF configuration so that DOF knows which output controller port is connected to each button lamp (the Start button lamp, Launch Ball button lamp, etc).

What controls the lights?

A lot of new pin cab builders have a hard time picturing how the lighted buttons get wired for software control. The thing that's confusing is that these devices are both buttons *and* lights. You're going to have to wire the buttons to a button input encoder device... so does that mean that the input encoder also controls the button lights? No.

The right way to think about this is to picture the button's switch and the button's lamp as completely separate devices. Then it's easy to see what connects to what:

- The switch connects to the key input encoder, just like any other button
- The lamp connects to the output controller, just like any other feedback device

This isn't just a convenient way of thinking about it, either. It's literally what's going on. The switch part and the lamp part are separate devices electrically. They're not connected to each other in any way in terms of wiring. They just physically occupy the same plastic housing. That makes it *look* like a single combined device, but it's really two devices inside one piece of plastic.

RGB or plain white?

Quick summary: plain white for the front panel buttons; RGB for the flipper and Magna Save buttons.

The front-panel buttons (Start, Exit, Launch Ball, etc) typically use plain white lamps, because the DOF config tool only provides monochrome settings for them. You *can* install RGB-capable LED lamps for these if you wish, but most people don't bother because of the lack of DOF support.

There's nothing technical stopping DOF from supporting RGB for the front panel lights. The only reason it doesn't is that there hasn't traditionally been any demand for it. This could change one day if enough people thought it was desirable.

DOF does provide full color control for flipper and Magna Save button lights, so if you install those, you should use RGB LEDs.

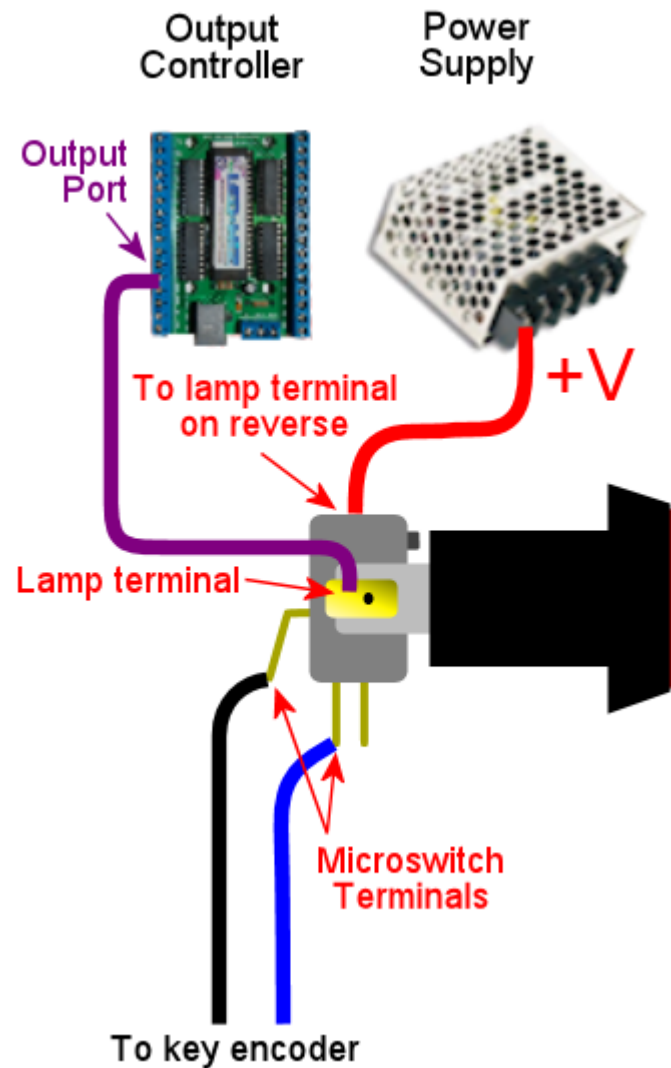
There's one other button where you might want to install RGB illumination: the lockbar Fire button, if you have one. DOF doesn't have RGB support for the Fire button lamp as of this writing, but there's at least one real machine that uses RGB lighting for it (Stern's *Star Trek*, 2013), so it seems like one that DOF should add in the future. Installing an RGB LED in your cab would be good future-proofing. You could also use it immediately by setting it up as a "Custom RGB" device in DOF, but that would require some extra work on your part to program the DOF Config Tool with the desired effects for each game.

(What does it actually mean for DOF to support RGB vs monochrome for a given output? In terms of hardware, full-color lighting is accomplished with a three-component LED: an LED with individually controlled elements for red, green, and blue light. The three color elements have to be controlled separately, so they require three output ports on your output controller for the single lamp. For example, if you have an RGB flipper button, your flipper button requires three separate output controller ports: "red flipper light", "green flipper light", and "blue flipper light". To use this with DOF, DOF has to think of the "flipper light" output as consisting of those three separate channels, so that it can generate the separate signals to control the relative brightness of the three ports. The DOF software itself is indifferent to this; it just thinks about the individual output ports. But the DOF Config Tool *does* know about it. The Config Tool has special handling for RGB devices that groups three consecutively numbered ports together into an RGB group, so that "RGB flipper button" really translates into the three output ports on your controller: red flipper button, green flipper button, blue flipper button. The Config Tool is inflexible about this; some devices are pre-ordained as RGB and the rest aren't. There's no way for you as a user to change that; it's up to the guy who runs the Config Tool. So when I say that "DOF" doesn't support RGB for the front panel lights, what I really mean is that the Config Tool doesn't have RGB definitions for the front panel lights. It assigns them as single-port = monochrome devices.)

Basic wiring plan

Keeping in mind that we're really working with two separate devices combined into

one housing - a lamp, and a switch - the wiring plan is simply a combination of the generic wiring plan for any output device (the lamp) and the generic wiring plan for any button (the switch).



Start, Exit, Extra Ball

These buttons are typically installed on the front panel, usually on the left side of the machine.

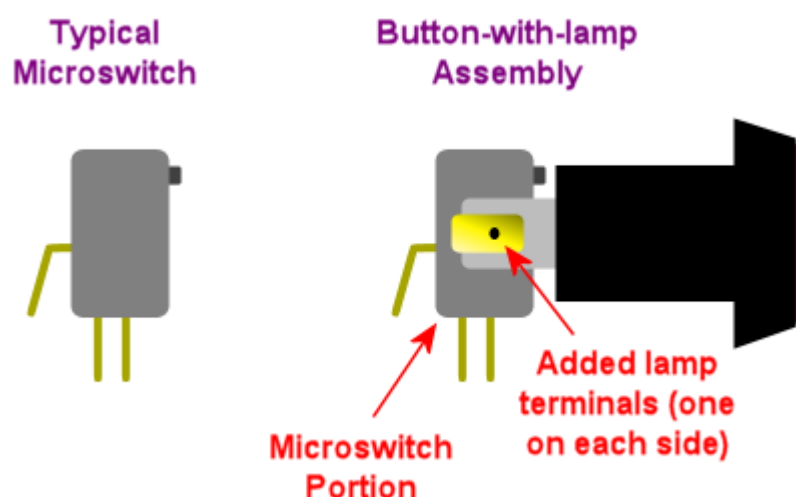
The standard part used on real machines (from the 1990s and later) is known as a "small round pushbutton with lamp", made by SuzoHapp, part number D54-0004-2x (the last digit is a color code). If you're searching at a pinball parts vendor (Pinball Life, Marco Specialties), look for Williams/Bally reference number 20-9663.



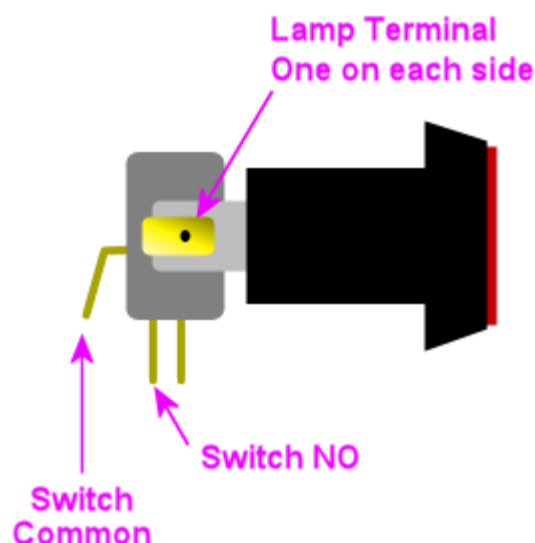
These buttons are socketed for #555 incandescent bulbs, which run on 6.3V and require about 250mA. If you buy from a pinball parts vendor, the button will usually come with a #555 bulb installed. The pinball parts vendors also sell plug-and-play LED replacement bulbs that fit the same socket.

Labels: If you buy your buttons from a pinball parts vendor, you can find buttons that come ready-made with labels for "Start", "Extra Ball", and a few other options. You can also buy blank buttons (with no labels) and install your own labels. You can print a label with any custom text you want using a laser printer and transparency film. To install a new label, you have to pry the plastic lens off of the button, remove any label already installed, insert your new label, and snap the lens back on. This is described in more detail in Chapter 34, Cabinet Buttons.

Identify the terminals: The first time you look at one of these button-plus-lamp assemblies, it can be a little daunting, because they have so many prongs sticking out. It's easier if you keep in mind that the base of the assembly is really just a standard microswitch, with a lamp socket bolted on top:



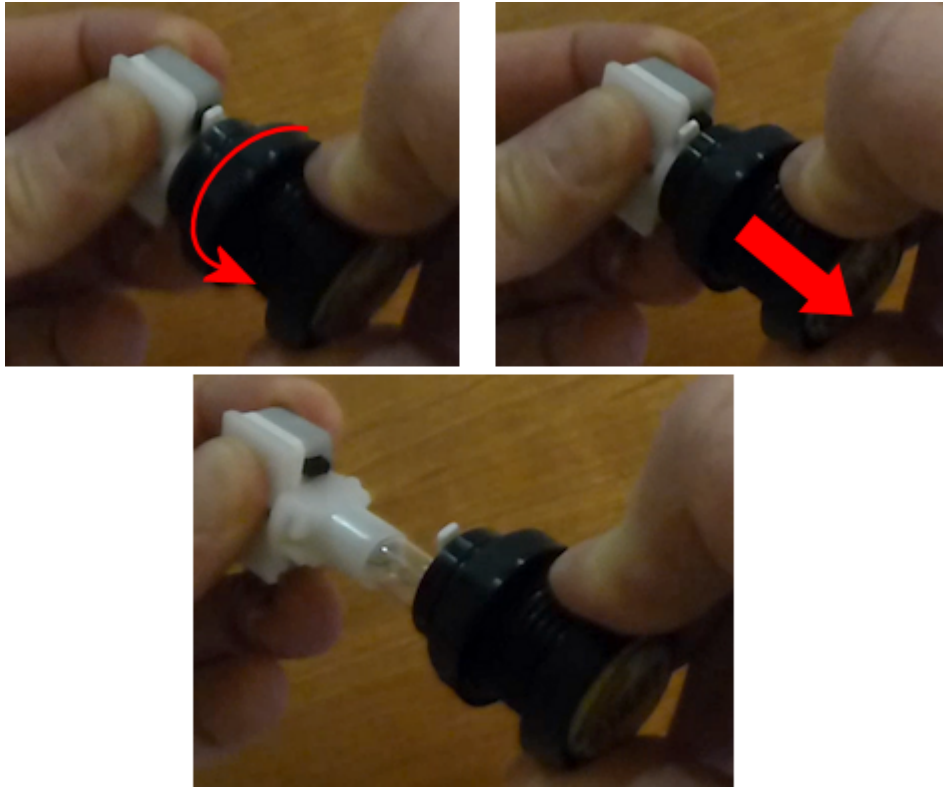
When you look at it that way, it's pretty easy to pick out the terminals for the switch portion and the added terminals for the lamp portion.



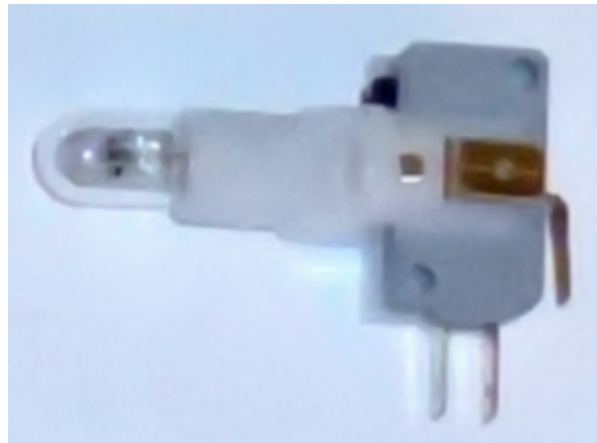
Note: the "Common" and "NO" (Normally Open) might not be positioned exactly as shown above for your switch. Double-check the markings on the switch to be sure before connecting anything. See Chapter 35, Button

Wiring for tips on reading switch markings.

This is even easier to see if you take the button housing off. To do this, gently twist the bottom switch portion slightly until it comes loose, then pull the switch housing away from the base.



Now you can see more clearly the way the lamp is tacked on to a normal microswitch.

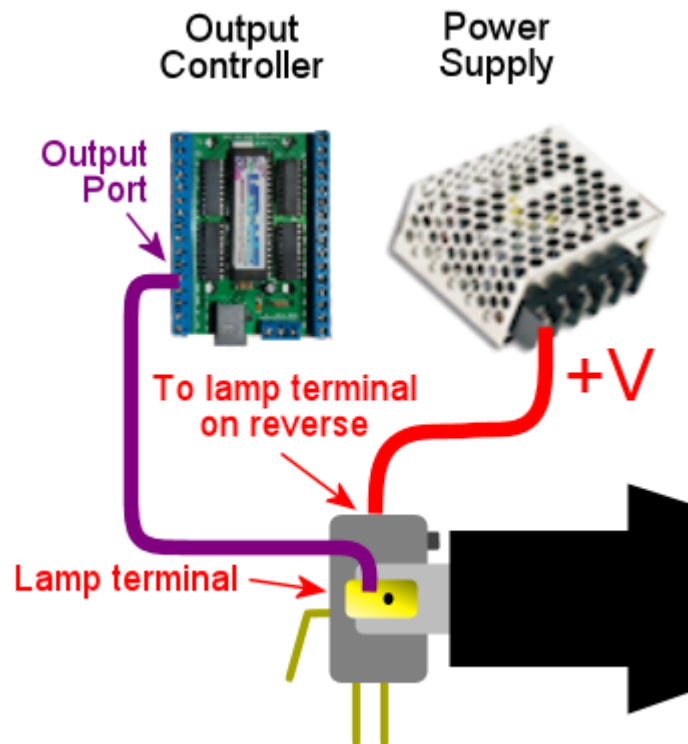


Wiring: Once you've identified the two terminals for the lamp, connect them to your output controller following the basic wiring plan above.

Incandescent bulbs aren't polarized, so there's no "+" or "-" to worry about; the two terminals are interchangeable. (LEDs *are* polarized as a rule, but nearly all of the #555 replacement bulbs are specially designed to work with either polarity, since they're designed to replace incandescents in existing machines.)

Connect one lamp terminal from the button directly to the "+" voltage from the power supply. You can use 5V, but for the incandescent bulbs, it's better to use 6.3V, which is the voltage they're designed for. See below for more on that.

Connect the other lamp terminal from the button to an available port on your output controller. Each button lamp requires its own separate output port, so that each lamp can be individually controlled.



DOF setup: The DOF Config Tool has entries specifically for the common front-panel buttons. On the Port Assignments page, assign the appropriate ports to the function that matches the button type:

- Start Button
- Extra Ball
- Exit
- Fire Button

See Chapter 34, Cabinet Buttons for more detailed descriptions of the button types.

6.3V supply: Incandescent #555 bulbs are designed to run on 6.3V. That's not a voltage you'll find on a PC ATX power supply, so many pin cab builders use the 5V ATX output, since it's reasonably close. That works, but the bulbs will be noticeably dimmer at 5V than they're meant to be, which might make your button illumination look a little sad. If you're using LED lamps, on the other hand, the voltage won't noticeably affect the brightness, so 5V is fine.

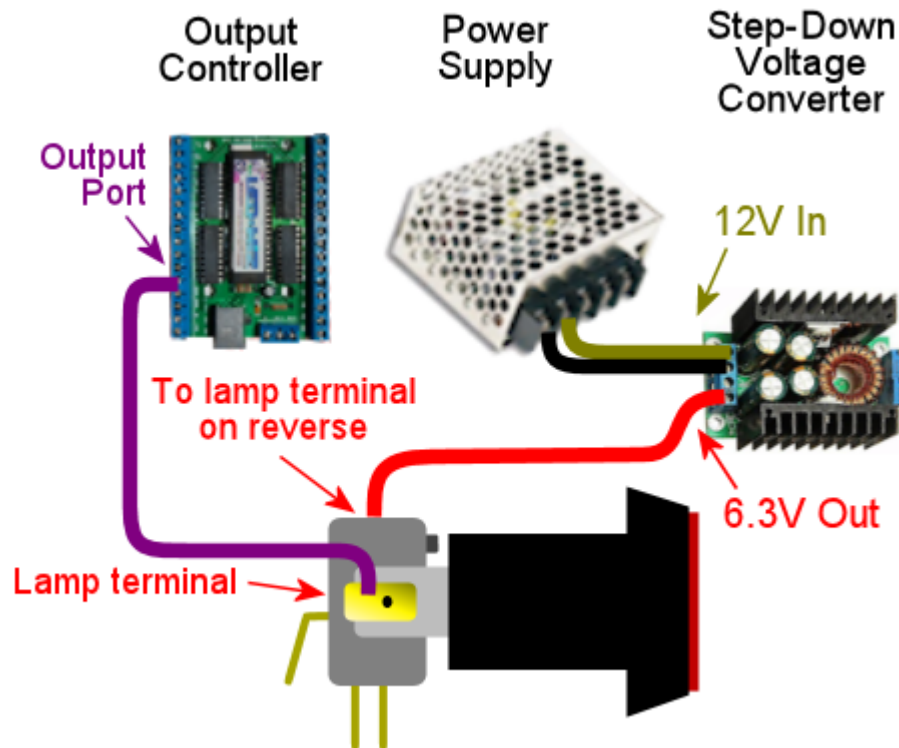
If you want to add a 6.3V supply, an easy way is to buy a "DC-to-DC step-down voltage converter". These are inexpensive (under \$10 on eBay) and easy to set up. If you buy one on eBay, look for the type with a variable output voltage, so that you can dial the desired 6.3V. Alternatively, pololu.com offers a high-current converters with a 6V output, which is close enough; that's a little easier since it doesn't require adjusting the voltage.

To connect:

- Connect 12V (yellow wire) and Ground (black wire) from your secondary ATX power supply to the step-down converter's (+) and (-) input terminals
- If your converter has an adjustable output voltage, adjust it to 6.3V, using a

voltmeter connected to the output terminals to measure it (you'll need to apply the power during this step)

- Turn off power
- Connect the converter's (+) output to one terminal of each 6.3V #555 lamp
- Connect the other terminal of each lamp to an available port on your output controller (following the standard wiring plan in Chapter 47, Feedback Device Wiring)



Launch Ball button

Several real machines from the 1990s used big round "Launch Ball" buttons in place of plungers. Many pin cab builders use the same buttons, either instead of plungers or in addition to plungers. See Chapter 37, Plunger for more on choosing whether to include one or the other or both, and where to place the Launch button if you're including one.

The standard type of Launch button is just a variation on the "small round pushbutton" type used for the Start button and the others above. The typical type is SuzoHapp part number D54-0004-1x (the last digit is a color code), which you can find at pinball parts vendors under Williams/Bally reference number 20-9663-B-4. The generic SuzoHapp type doesn't have any label text; the type sold by pinball parts companies will include a "Launch Ball" label insert.



Wiring: the Launch Ball button is wired exactly like the small round pushbuttons described above.

DOF setup: The DOF Config Tool has several entries specifically for Launch Ball buttons:

- Launch Ball: choose this if your cab has *only* a launch button, and no plunger.
- Authentic Launch Ball: choose this if your cab has both a plunger and a launch button.
- ZB Launch Ball: ignore this one. It's not for button lamps, but for something else entirely (see Chapter 109, ZB Launch Ball).

Why the distinction between Launch Ball and Authentic Launch Ball? It's to accommodate different styles of cab building and different personal tastes.

The Authentic Launch Ball is designed to replicate *exactly what the original pinball machine would have done* in terms of lighting up the Launch button. It's controlled purely by the emulated ROM, just like in the original machines. That means that it *never* lights up at all when you're playing a table that originally used a plunger rather than a Launch button. This is the right choice if you have both a plunger and a Launch button, since the Launch button will only draw attention to itself when you're playing a game that actually had a Launch button in the original version; it'll stay dark when you're playing a table that originally used a normal plunger.

The Launch Ball button, in contrast, lights up much more often - essentially whenever there's a ball in the launch chute. This isn't true to the original games, since most original games didn't have a Launch button at all. But if your pin cab doesn't have a plunger, you'll probably prefer this, since it makes every game (even games that originally had plungers) act like the Launch button is part of the game.

Big rectangular buttons

Some pin cab builders use a larger rectangular version of the buttons from the same family as the "small round pushbutton" type above. You can find these under SuzoHapp part number D54-0004-5x (the last digit is a color code). SuzoHapp and other arcade suppliers sell a number of other version of basically the same button in different sizes and shapes, so you should be able to find something to match the look you're going for with a little searching.

These buttons all work just like small round pushbuttons above in terms of wiring.

Flipper buttons

You don't often see this on real pinball machines, but a nice embellishment for virtual cabs is to illuminate the flipper and Magna Save buttons.

The DOF Config Tool has support for this. A really nice touch is that the Config Tool database is pre-programmed to use RGB coloring to match the original arcade button colors for most tables. Whenever a table is loaded, the flipper buttons will light up in a color matching the plastic button color in the original arcade version. If you have Magna Save buttons, those will light up for games that originally had similar extra buttons, also using the same colors as the original arcade machines.

Virtually every real arcade pinball machine had matching colors for the left and right flipper buttons, so the DOF config tool doesn't distinguish between left and right. When you're planning your wiring, you should simply connect the left and right

flipper button lights to the same output ports. In contrast, machines with extra flipper buttons often were asymmetrical about it (usually, the asymmetry was simply that an extra button was installed on one side only rather than on both sides), so the DOF config tool does distinguish between left and right Magna Save outputs. If you can spare the additional output controller ports, you should wire the left and right Magna Save buttons to separate ports, so that they can be controlled independently.



Be sure to use RGB LEDs for these lights, so that DOF can control the color.

Parts: For full-color flipper button lights, you need either a clear plastic button or a translucent white plastic button. I much prefer the completely clear type because they're a lot brighter, and they look more interesting to me. Some people worry that clear plastic wouldn't diffuse the light enough to make the light visible, but in fact they're really very bright, and the internal reflections in the plastic make for a pleasing jewel effect. (The picture above shows my clear plastic buttons. This is with the camera flash on, so you can get a sense for how bright they are.) Translucent white plastic produces more diffuse light but reduces the brightness.

The clear plastic flipper buttons are available in 1 $\frac{1}{8}$ " and 1 $\frac{3}{8}$ " lengths:

- Standard 1 $\frac{1}{8}$ " length, Williams part number A-16883-13
- Longer 1 $\frac{3}{8}$ " length, Stern 515-7791-00 (this is nominally sold as the "Fire" button for Stern lockbars, but it's really just a standard flipper button in the longer length)

If you bought a button kit from Virtuapin, it probably included the longer 1 $\frac{3}{8}$ " buttons in opaque red and white, with some neat little plastic leaf switch holders that attach directly to the buttons. Those switch holders only work with the longer 1 $\frac{3}{8}$ " buttons, so be sure to get the long buttons if you want to use the switch holders in your final setup.

The shorter buttons are cheaper and easier to find in clear plastic, but you can't use them with the Virtuapin switch holders. The switch holders are convenient but are hardly required - it's really not all that hard to mount the leaf switches without the holders. On my own cab (and on all of the "real" pinball machines I've seen), they're simply attached directly to the side wall of the cabinet. You'll need some kind of spacer to get the positioning right, but otherwise it's easy. See the photo below of my leaf switch mounting to see how this looks.

For the lights, there are a few options:

- DIY, using my custom circuit board design. You can find my EAGLE plans for a board that works like the Lightmite boards here:

mjrnet.org/pinscape/downloads/Flipper_Button_LED_board.zip

It's easy to manufacture those boards. Just go to OSH Park and upload the **.brd** file from the download, and click through to the order page.

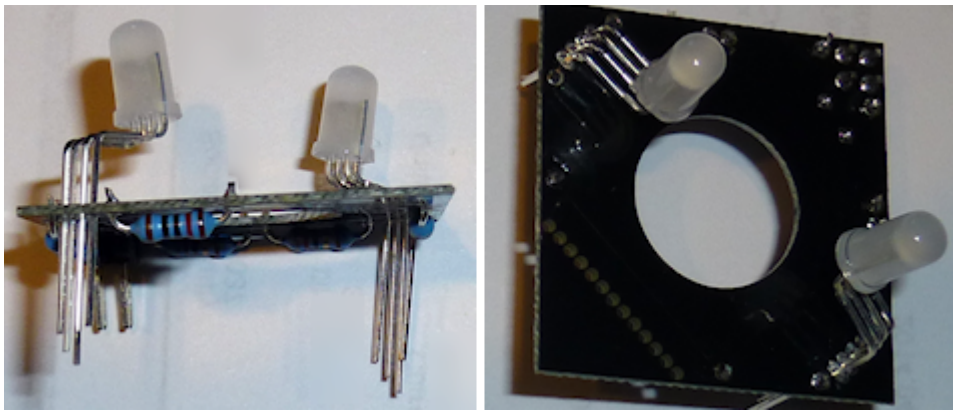
My EAGLE plan are designed to be drop-in replacements for the Lightmite boards, so to assemble them, follow the same instructions below for the

Lightmite boards.

- Oak Micros's flipper button LED boards (no longer available). These come fully assembled with the LEDs built in, and they're compatible with the VirtuaPin leaf switch holders as long as you use the longer 1-3/8" flipper buttons. As of June 2021, I don't think Oak Micros is selling any of their products any more, but you might check the original vpfforums thread for any updates:

www.vpfforums.org/index.php?showtopic=43571

- Lightmite LB boards from NiceMite.com. I used these for my own build and would be happy to recommend them, but unfortunately they don't seem to be available any more. The site is down as of this writing (April 2020), and although it says "Temporarily Closed", I'm pretty sure it's really gone for good because it's said that for at least six months. If they should ever come back, go with either the plain board or the assembled kit with the **manually controlled 4 lead RGB LED**. (**Don't** buy the "auto changing" kits. Those don't allow software control over the colors.)



The assembled Lightmite boards. The LEDs stick out on the back. The leads should be bent so that the LEDs will fit into the open space around the button.



Installing the Lightmite boards. Upper left: the inside of the cabinet with the clear plastic flipper button installed. Upper right: positioning the Lightmite board. Lower left: the board in place. Lower right: with the pal nut installed, screwed onto the shaft on the back of the button.



Full Lightmite board installation, with the leaf switch back in place. Note that I used a small piece of plywood as a shim to get the leaf switch positioning right. The leaf switch has to be set off from the cabinet wall by about 1/4", so you'll probably need a similar improvised spacer.

Assembly instructions for Lightmite boards or my EAGLE plans: Buy one board/kit per button. You need two boards/kits if you only have a pair of flipper buttons, or four if you have both flipper and Magna Save buttons.

If you buy the plain Lightmite board or use my EAGLE plans, you'll need to buy the LEDs and other electronics to populate the board. Buy the quantities listed below **multiplied by** the number of boards you're assembling.

- (2) LEDs: **common anode** (common positive lead) 5mm, 20-25mA RGB LED, such as Kingbright WP154A4SEJ3VBDZGW/CA
- (2) resistors for the red channel: for the LED above, 150 Ω 1/8W
- (2) resistors for the green channel: for the LED above, 100 Ω 1/8W
- (2) resistors for the blue channel: for the LED above, 100 Ω 1/8W
- (1) 2x2 0.1" pin header
- (1) 2x2 0.1" female crimp pin housing with 4 crimp pins

The pin headers and crimp pin housings are the same types used in several places in the expansion boards; see Chapter 91, Electronic Parts List and Chapter 82, Crimp Pins.

Wiring: Connect the pins on the Lightmite boards or my EAGLE boards as labeled:

- Connect the **+** pin on the Lightmite board to +5V from your secondary ATX power supply

- Connect the **R**, **B**, and **G** pins from the Lightmite board to ports on your output controller. Each color pin connects to one port. **The ports must be consecutively numbered, and must be in R-G-B order.** For example, if you connect R to port #15, connect G to port #16 and B to port #17.
- If you have enough ports to spare, connect the left and right flipper buttons to separate port triplets so that DOF can control them separately. If you want to conserve ports, you can simply connect the left and right flipper buttons together. DOF doesn't currently distinguish left and right flipper button colors, so there's no immediate benefit to giving them separate port assignments, but connect them separately anyway if you can so that you'll be able to take advantage of any future updates if DOF does eventually separate them.
- Connect each Magna Save button (left and right) to its own group of three output ports. You can connect them together if you want to conserve ports, but separating them allows DOF to control the left and right sides separately.

DOF Setup: Assign the red/green/blue port group for the flipper buttons to the "RGB Flippers" device type in the DOF Config Tool. If you wired the left and right buttons to separate port groups, assign both port groups the same way, both to "RGB Flippers". DOF doesn't provide distinct left/right versions of the flipper lights, because essentially every real pinball machine used matching colors for the left and right flipper buttons in the original arcade version.

DOF does have separate left/right versions of the Magna Save buttons, because there are many examples of real machines where the left and right "extra" buttons weren't symmetrical. In nearly all cases, the difference is that an extra button was only installed on one side. In these cases, the separate port assignments let DOF leave the missing button unlit, to better replicate the original arcade setup. So in the DOF Config Tool, assign the port group for your left Magna Save button to "RGB Left Magnasave" and connect your right button ports to "RGB Right Magnasave".

Coin chute buttons

Most people leave the coin chute lights hard-wired to constant power. If you have a separate Coin button, you can do the same thing. Even in the real machines, the coin chute lights are almost always hard-wired to stay constantly on. You *can* wire these to DOF outputs if you really want to, and DOF provides a "Coin" output type, but there's not much practical benefit, as the standard DOF Config Tool database setting simply leaves this output constantly on, just like the real games.

If you're using a real coin door, you can find more information about how the coin chute lights are wired (and how to access that wiring) in Chapter 40, Coin Door.

56. Flashers and Strobes

Flashers and strobes are extra lights on your cab to reproduce the bright flashing light effects of the real machines. I personally rank flashers in the top tier of must-have feedback devices, as they add a huge amount of excitement and realism to the playing experience.

Pinball machines have always used blinking lights to attract players and add to the visual appeal of the game. The light shows became more sophisticated when software started controlling the games in the 1980s and 90s, as complex lighting patterns could be programmed and coordinated with the game action.

One of the key light show components in the 1980s and 90s machines is the "flasher" light. A flasher is a bright lamp in a plastic dome, usually situated above the playfield, maybe near a ramp or other vertical element sticking up out of the playfield. Games from the 80s and 90s typically have five or six flashers distributed around the playfield. As the name implies, flashers light up briefly at key times to create a brilliant strobe effect, to underscore an event in the game and add drama.



Flashers in real pinballs: alongside the right ramp in Earthshaker; behind Rudy in Funhouse

The flasher lights from the original games are, of course, reproduced on-screen in the Visual Pinball versions and other simulator versions. They appear in the video version just where they would have been on the real playfield, and they light up when they're supposed to. That's fine for desktop play, but when you blow it up to life size in a pin cab, it becomes more apparent that they're artificial; the video version just isn't as bright as the real thing.

That's why many pin cab builders include actual flasher lights on their cabs. Rather than relying entirely on the on-screen video renditions of the flashers, you can supplement the video display with a set of real flashers, which produce the same brilliant light as the real thing.

Related to the flashers is the "strobe" light. This is an embellishment thought up by the pin cab community, not something you find on real pinball machines (not commonly, anyway). It usually takes the form of a very bright white light somewhere in the player's line of sight. The DOF software will flash this briefly (thus the "strobe" monicker) at dramatic points in the game play. This might just seem like one more flasher light, and in broad terms, it is. The only reason for calling it out as a separate device is that it's usually implemented with a much brighter light source than the regular flashers, so that it produces a particularly dramatic burst of light at appropriate moments.

Flasher panels

In the real games, flashers are arranged around the playfield in different places in each game. There's obviously no way a virtual pin cab can reproduce the exact placement of the original flashers in every game (other than with the video renditions that we've already decided we want to supplement). We can't even move them around per game; we have to pick a fixed set of locations. So the usual arrangement is to group a set of flasher lights together in a "flasher panel", usually arranged in a straight line across the width of the cabinet.

There's no fixed rule about how many flashers you use or where to place them. It should go without saying, but it's your cab, so it's your call. But given the constraints, there are some common arrangements that most cab builders end up using. Let's look at the options.

Number of flashers: Most pin cab builders use a panel with three or five flashers.

What's magical about the numbers 3 and 5? History, mostly. The DOF Config Tool database has specific support for 3- and 5-flasher setups, because those are the most common setups that people build. (And now everyone goes with those setups because they're what's supported.) The 5-flasher setup also has a somewhat deeper significance in that it maps well to the original physical flasher layout in most real tables. Most of the 1980s and 90s machines used about five flashers total, so a 5-flasher panel lets DOF map the flasher effects from the original games directly to your physical lights.

You can build panels with any number of flashers you want, but a 3- or 5-flasher setup is easiest to work with because of the pre-programmed DOF support.

I'd recommend the 5-flasher setup if you're building a full-size cab, since it will easily fit and will produce the most interesting light show effects. The 3-flasher setup is mostly for mini-cabs where there's not room for five.

Placement: Once you have your panel of three or five flashers, the possibilities for where to situate it become fairly apparent based on the space available in your cabinet. The most common places are:

- At the back edge of the playfield TV. This is the most common placement, because it answers the question "what should I do with the extra space behind my TV?" that most cab builders are faced with. Nearly all pinball cabinets designs (even the "widebody" plans) are longer and narrower than a standard 16:9 TV, so there's usually some extra front-to-back space to fill, typically between 3" and 6".

You can position the panel so that it lies flat in the same plane as the TV, or you can tilt it up at an angle, if you left any vertical space between the playfield and the glass cover. My panel is tilted up at about a 30° angle to fill the vertical space (see the photo below).

- Vertically, on the inside back wall of the cabinet above the TV. Some people prefer this arrangement because the lights point more directly at the player, which might slightly reduce reflections from the glass cover.
- On the front of the backbox, in the speaker panel area just below the DMD. If you positioned your TV at the very top of the cabinet with no room between the TV and glass cover, you probably can't fit flashers inside the cabinet. You can position them outside the cabinet instead, on the front of the backbox.
- Across the top of the backbox, near the front edge. I have an extra set of five flashers here on my cab. This is a good place for a supplemental set of flashers, but I wouldn't recommend it as the location for a lone set, since it's not in the player's line of sight. That greatly reduces the impact. It does add to

the effect for bystanders, though, which is why I like it as a secondary location.

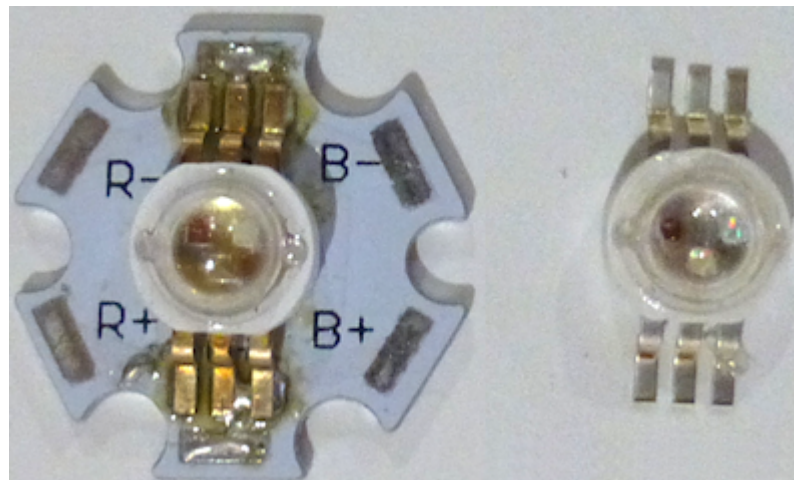


My flasher panel. Five flasher domes are distributed across the width of the cabinet. The panel is at the back of the playfield TV, tilted up at about a 30° angle to the plane of the TV, so that it fills both the horizontal and vertical gap at the back of the cabinet.

Building a flasher panel

LEDs: For the full effect, you'll want to use an RGB LED for the flashers. This allows the software to control the color emitted. An RGB LED has separate elements internally for red, green, and blue light (the "R", "G", and "B" of the name).

The type that most people use in pin cabs can be found on eBay by searching for "star base LED" or "3W LED RGB". They look like this:



The type on the left is the "star base" type. These come with a metal base that serves as a heat sink and makes the wiring a little easier. The type on the right is actually the same LED, just without the base. You can buy them either way. Most people think the type with the base is a little easier to work with.

These are sold in individual colors as well as RGB, so be sure to order specifically the RGB type. All of the parts I've seen for these on eBay are the same basic device, so there's not too much risk of finding a wrong part that looks similar, but here are the approximate electrical specs to look for just to be sure you have the right thing:

- Red channel: 2.2 to 2.4V, about 400mA
- Green channel: 3.2 to 3.4V, about 400mA
- Blue channel: 3.2 to 3.4V, about 400mA

Plastic domes: Most pin cab builders use the same domes used on the real pinball machines for their flashers. They come in two main types, which are different only in how you attach them. Here are the Williams part numbers:

- Screw tab style (03-8149-13)
- Twist-on style (03-8171-13)
- Base for twist-on domes (03-8172-13)

The screw tab type is a little easier to install; the twist-on type looks a little cleaner since the fasteners are hidden under the dome. You can easily make either type work as long as you plan for it.

The domes come in various colors (red, purple, amber, etc.), but for pin cab purposes, you almost always want the clear type, since we use them with color-changing RGB LEDs. If you want a particular color for some reason, the last two digits of the part numbers above are the color code for "clear", so you can find the other colors by searching for the part number prefix without the -13 suffix.



Voltage supply: Plan on supplying the flasher LEDs from a **5V** power supply.

A lot of new cab builders find LED voltage specs confusing. LEDs always quote a "forward voltage", and it's always something weird like 2.2V or 3.4V. And it's usually a range, like 2.2-2.4V. Does this mean that you're supposed to find a 2.2V power supply? Fortunately not! The "forward voltage" for an LED **isn't** the supply voltage required. It's the amount of voltage that the LED "drops" when operating, which is sort of the amount it uses or consumes. That's why it's often quoted as a range; the manufacturer is just saying that the exact number varies a little bit from one individual part to the next, but it should always be in this range, and will usually gravitate to the middle of the range. What does this mean for the supply voltage, then? It's really simple: the supply voltage can be *any voltage higher than* the forward voltage. So if the forward voltage is "3.2-3.4V", it means that you can use pretty much anything higher than 3.4V. It's best to pick the lowest conveniently available voltage higher than the required minimum, because the higher the voltage, the more energy will get burned up in heating the resistors. In a pin cab, you usually have a 5V supply readily available, (see Chapter 45, Power Supplies for Feedback), so 5V is the best option for almost every LED. If you had a 4V supply or a 3.7V supply, that would be better still, but those aren't typical voltages in a pin cab; 5V is what we usually have readily at hand.

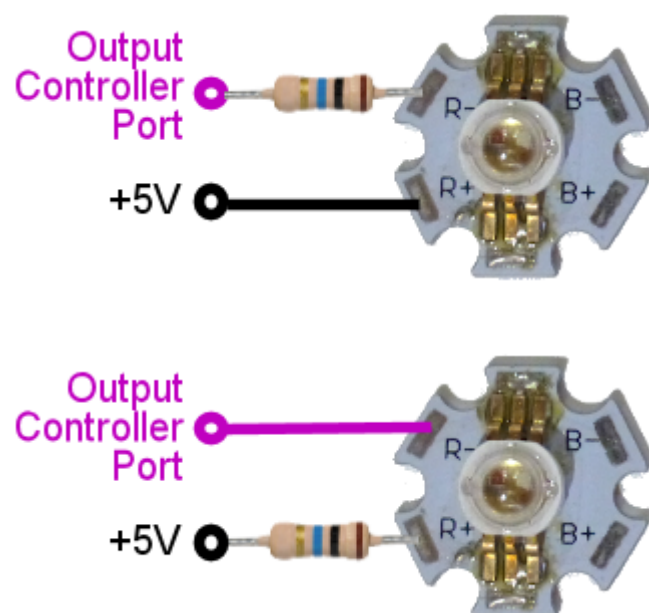
You might read advice on the forums to use a 12V power supply for the flasher LEDs. Ignore that. It's best to use the lowest conveniently available supply voltage above

the LED's "forward voltage" spec. Using a 12V supply just burns up all of the extra voltage in resistor heating, which forces you to use physically larger resistors that can tolerate all of the extra waste heat. A 5V supply lets you use smaller resistors that waste less energy.

Resistors: The star base LEDs are bare LEDs that **do not** include any sort of built-in current regulation. This means that you **must** include separate resistors in the circuits when wiring them.

It's important to choose the correct type of resistor for each color channel. Chapter 52, LED Resistors explains how to choose, and includes a calculator to determine the correct Ohms and Watts value for each.

The resistors have to be wired into the circuit in series with the LED. They can go on the (+) side or the (-) side; it's exactly the same either way.



You can wire the resistor for the LED on the negative side or positive side, whichever is more convenient for your setup. It's the same electrically either way.

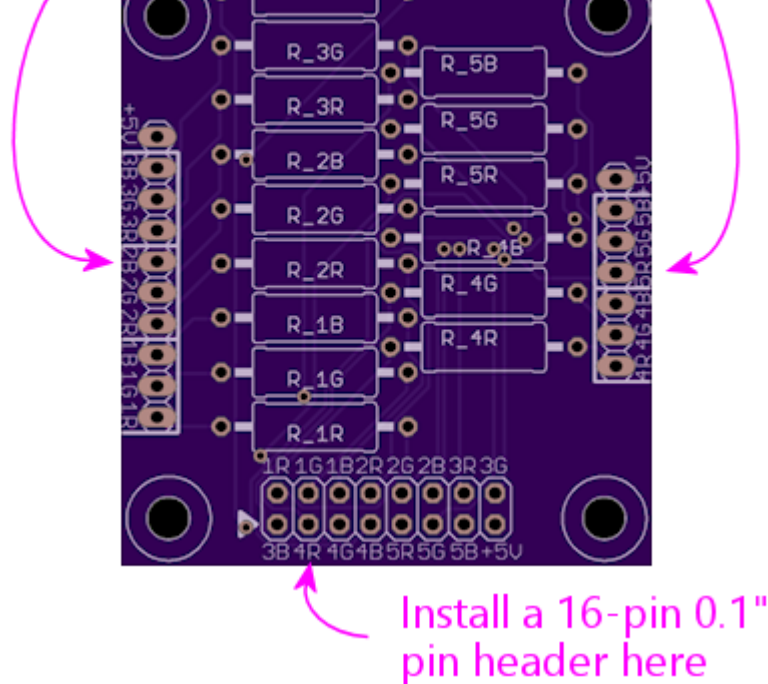
The red, green, and blue channels are independent devices, so each channel needs its own resistor. The channels will have different resistance values, too; that's why they quote the "forward current" and "forward voltage" separately for each color.

Pay attention to the wattage value that the resistor calculator reports, and be sure you buy resistors with at least the required wattage. This is **not** some kind of "detail for nerds" that you can ignore; it's part of the spec for what you need to buy.

Here's a simple circuit board design for the resistor panel. This is designed to be used with a ribbon cable connector to the output controller board. The pin layout of the 16-pin connector (at the bottom of the board as pictured below) matches the pin layout of the Pinscape expansion board RGB Flasher connector, so you can conveniently connect this directly to the expansion board with a ribbon cable. You can also easily use this board with non-Pinscape output controllers; you just have to program the controller so that the LED channels match the pins on the connector.

Solder hookup wires
to LEDs, daisy-chain the +5V





Here are the EAGLE plans, if you want to have this board manufactured:

mjrnet.org/pinscape/downloads/flasher-resistor-board.zip

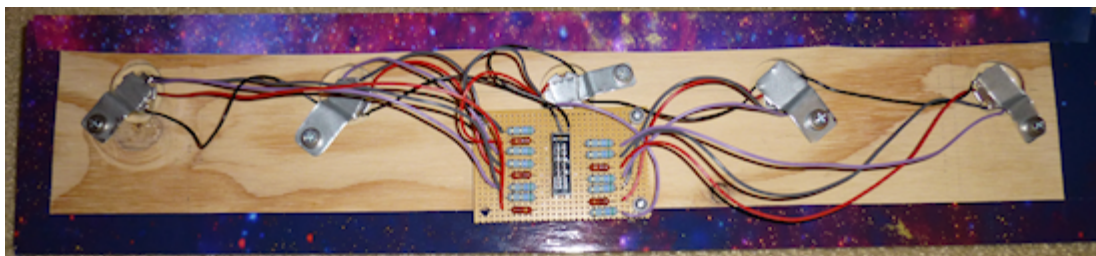
You can have it made in lots of three by OSH Park for about \$16, with an extremely simple ordering process (just upload the .BRD file from the ZIP above). You can have it made more cheaply per board at PCB makers like Elecrow, but they have larger minimum order sizes and require you to generate Gerber files. That procedure is explained in Chapter 93, Fabricating the Expansion Boards.

To assemble this board:

- Install a 2x8-pin 0.1" pin header as shown in the picture
- If you use a shrouded header for the 2x8 connector, make the sure the keying slot on the shroud is aligned properly for your ribbon cable; see Chapter 81, 0.1" Pin Headers
- Figure the resistor values required as explained in Chapter 52, LED Resistors
- Be sure to pay attention to the wattage calculation in that section, and use resistors with at least the required wattage specs
- The board is designed for resistors **up to 4mm in diameter** and **up to 12mm in length** (the gigantic squarish "cement" resistors **won't** fit)
- Install the resistors in the R_xx slots: R_1R is the resistor for LED 1 RED channel, R_2B is LED 2 BLUE, etc
- Resistors aren't polarized, so it doesn't matter which direction you install them
- I'd solder hookup wires from the boards to the LEDs directly to the edge connector terminals; 1R is for LED 1 RED, 2B is LED 2 BLUE, etc
- If you prefer, you can install single-row 0.1" pin headers on the edge connectors, but I don't think that's necessary in this case, because I'd consider this board to be an integral part of the flasher board that can be hard-wired to the LEDs
- If you're using the Pinscape expansion boards, simply connect the resistor board to the RGB Flashers header using a 16-conductor ribbon cable with a 16-pin IDC plug at each end; see Chapter 83, Ribbon Cables for help building these cables

- When plugging in the cable, be sure that the ribbon cable is oriented so that Pin 1 on the expansion board connects to Pin 1 on the resistor board (Pin 1 on both boards is marked by the little triangular arrow next to the header)

Building the panel: I went with a simple design for my panel.



- I started with a piece of 3/8" plywood, cut to the inside width of my cabinet and a height that I measured to fit the available space where I planned to install it.
- I drilled five 1/2" holes at regular intervals, where I wanted to place the lights.
- I soldered four wires to each of the five star LEDs: one wire to each of Red(-), Blue(-), and Green(-), and a fourth wire that I connected to all of the (+) terminals. As described in Chapter 47, Feedback Device Wiring, the standard wiring plan for all devices is to switch on the (-) side, so the (+) side is always connected directly to power and can thus be daisy-chained across all devices that share the same voltage.
- I then installed the LEDs in the pre-drilled holes, holding them in place with little metal clamps. The clamps aren't anything standard; they're just something that I made custom by cutting them out from a piece of sheet metal from Home Depot.
- I ran the wires from the LEDs to a small circuit board containing the resistors (see above). This is just an ad hoc circuit board I made for the cab. You can just wire the resistors in-line through the wires without a circuit board if you find that easier.
- On the front side of the board, I attached the plastic dome with wood screws.

Finishes for the panel: Most people just paint their panels in the same color as the inside cab walls, usually black. I used custom-printed adhesive decals that I had made at the same time as the decals for the exterior of my cab. See Chapter 22, Cabinet Art.

Heat sinks: Some people use large heat sinks with these LEDs. I don't personally think it's worth the trouble, because flashers are activated intermittently, for brief periods. Plus, the metallic star bases already provide a moderate degree of heat dissipation. If you were leaving these LEDs on at full power for long periods, heat sinks would be warranted, but they're not necessary in this application.

If you want to use heat sinks anyway, you can find a variety of suitable products on eBay. Attach them to the back of the star base with thermal paste.

Wiring to the output controller: Follow the general plan in Chapter 47, Feedback Device Wiring to connect the LEDs to your output controller. Connect the (+) side of each LED to the +5V from your power supply, and connect the (-) side to an available port on your output controller. Keep in mind that each channel requires a resistor (see above) in series with the LED, inserted either between +5V and the LED (+) terminal, or between the output controller port and the LED (-) terminal.

The DOF Config Tool requires that you use **consecutively numbered output ports** for each Red-Green-Blue group, **in that order**. For example, if you wire the RED channel of Flasher #1 to output port #8, you must wire GREEN to port #9 and BLUE to port #10.

DOF setup: In the DOF Config Tool, after you've done the basic setup for your output controller, go to your Port Assignments page. For each flasher LED in your setup, find the port number for that device's RED channel, and set it to the appropriate flasher device:

- For a 5-Flasher setup, the flashers are labeled, from left to right, "5 Flasher Outside Left", "5 Flasher Left", "5 Flasher Center", "5 Flasher Right", and "5 Flasher Outside Right".
- For a 3-Flasher setup, the flashers are labeled, from left to right, "3 Flasher Left", "3 Flasher Center", and "3 Flasher Right".

When you select the appropriate "5 Flasher" or "3 Flasher" device for the RED channel's port, the Config Tool will automatically assign the next two output ports in sequence to the GREEN and BLUE channels for the same device. This is why you had to physically wire each LED to three consecutively numbered ports in Red-Green-Blue order. It's just the way the Config Tool thinks about RGB, so you have to do the wiring to match its expectations.

Pinscape expansion boards: If you using the Pinscape expansion boards, you should connect the flashers to the dedicated Flasher outputs on the main board. These outputs have enough power handling capacity for about three sets of flashers in parallel, if you have multiple sets (such as my arrangement with the main flasher panel at the back of the cabinet, plus a second set on top of the backbox).

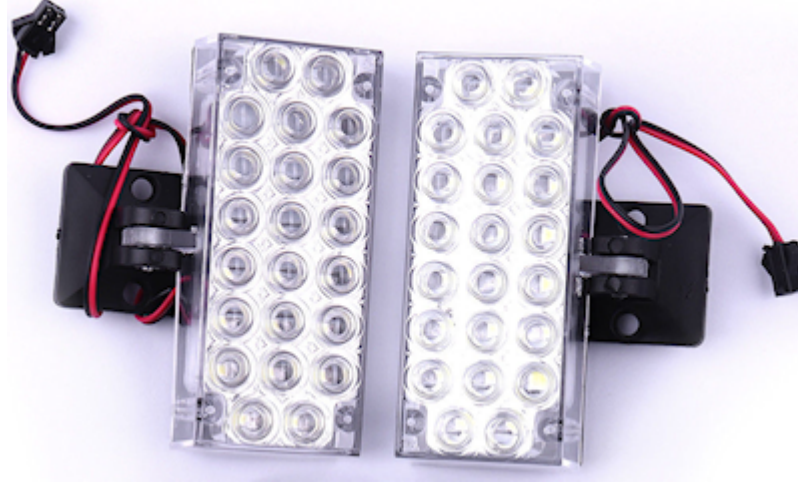
Strobes

Strobes make a nice accompaniment to flashers. Strobes are simply bright white lights, usually much brighter than the flasher lights, that the software can light up occasionally for particularly dramatic events in the game.

Unlike flashers, strobes are typically monochrome, not RGB. They're just plain white lights. This makes them a little quicker and easier to set up since you only need a single output port to control them.

One question that new cab builders always ask is whether you need some kind of special timed bulb that handles the rapid flashing effect. The name "strobe" certainly suggests this. The answer is no: the flashing effect is handled entirely by the DOF software. All you need for the physical device is a bright white LED.

Parts: The device that most pin cab builders use for strobes are small but bright LED panels designed for automotive use (emergency lighting, spotlights, etc). They can be found on eBay by searching for **22 LED white strobe**. (This search might also turn up some "22 **Inch**" LED light bars for trucks. Ignore those. You're looking for the smaller "22 **LED**" products.) The ones you're looking for look like this:



Since these are designed to be installed in cars and trucks, they usually come with a separate control box, which lets you blink the lights in various flashing patterns. You **don't** need these control boxes for DOF - just discard the control box and plug the LED directly into your output controller. That will let DOF control the flashing effects directly. (This also means it's perfectly fine to use an LED that doesn't come with any sort of control box.)

Positioning: A lot of people put a pair of the 22-LED strobes on their flasher panel, usually one at each outside edge. Others put them on top of the backbox or in the speaker panel area.

I put mine on top of the backbox, facing forward (facing the player). I'm happier with that location than in the main flasher panel because I find them too bright to have directly in your line of sight while playing. Having them on top of the backbox makes the strobe effect suitably dramatic without blinding the player every time they fire.

Power supply: Check the specs on your specific product to be sure, but the standard power requirement for strobe panels like the ones described above is 12V DC. The same is true for almost anything made for automotive use, since that's the standard car "cigarette lighter" voltage.

LED resistors: Not needed for the typical 22-LED panel. These are meant to be plugged directly into a car's 12V power supply, so they have the necessary resistors built in.

Wiring: Follow the basic wiring plan in Chapter 47, Feedback Device Wiring:

- Connect the 22-LED panel's (+) terminal (usually the red wire) to the +12V power supply (the yellow wire from your secondary ATX power supply)
- Connect the LED panel's (-) terminal (usually the black wire) to an available port on your output controller
- If you bought one of the car-and-truck strobes that comes with a blinker control box, discard the control box and connect the wires from the LED panel directly to your output controller (LedWiz/Pinscape/etc). DOF will handle the flashing effects, so you don't need any separate blinker control box; including it in the circuit will just interfere with the wider range of effects DOF can create.

No additional parts are needed with these (in particular, no resistors are needed).

The typical 22-LED panels require about 500mA. This is just barely within the safe range for an LedWiz output, so you can connect it directly to an LedWiz. **Don't** connect two panels in parallel to the same LedWiz port, though, as this would require 1000mA, which is double the LedWiz's safe limit.

Pinscape expansion boards: You can connect a 22-LED panel to the dedicated "Strobe" output, to any RGB Flasher port, or to a power board port. Any of these ports will handle two of the 22-LED panels wired in parallel.

DOF setup: In the DOF Config Tool, go to your Port Assignments page. Find the output controller port number where you wired the strobes. Assign this to "Strobe".

57. Beacons

In virtual pin cab parlance, a beacon is a rotating or flashing light like you'd find on a police car or ambulance. Beacons make good backbox toppers, since they're the right size, and they can play an active role in the light show. They're also a good simulation feature, since lots of real pinball machines featured beacon toppers.



Parts: The only part needed is the beacon itself, which you can buy as a ready-made product. I haven't seen any good DIY designs, probably because it's pretty easy to find off-the-shelf products that fill the bill.

The thing to look for is "police beacon" or "rotating beacon light". You can look for these on Amazon or eBay, but unlike most product categories, neither site tends to have very good options. The ones you'll find on either site are mostly "disco" or "party" lights - novelty toys, which won't be very bright or realistic. So instead of looking at a general retailer, you might try specialty automotive supply stores, such as Vehicle Safety Supply.

A particularly good option that I've found is the **Peterson 771** series. I use a pair of these on my cab. They're inexpensive (about \$25 apiece), they're exactly the right size and shape, they feature a traditional motorized rotating light with directional reflector, and they're available with red, blue, or amber lenses. They're a perfect fit for this job. To find them, search on the Web for **Peterson V771A** for amber, V771B for blue, and V771R for red.

Another option is the very similar looking Wolo Manufacturing 3100 series: 3100-A (amber), 3105-B (blue), 3110-R (red). I haven't seen these in person, but from the pictures and specs, they look essentially identical to the Peterson V771 series. They might even be the same thing under a different brand name. Current prices are about \$20 per unit.



Beacons mounted on my cab's backbox, one red and one blue. These are off-the-shelf items, Peterson model 771, available from automotive supply sites.

Mounting: Many of the beacons you'll find (including the Peterson 771) come with

magnetic bases that are intended to hold the light to the roof of a car. You'll probably want to remove the magnetic base and attach the light to the top of your backbox with screws or other wood fasteners. You could alternatively glue them in place, although I try to avoid glue as it can't be easily removed if you ever need to replace the device.

Wiring: Check with your product's documentation to be sure, but anything automotive generally runs on 12V DC. That works well in a pin cab because we usually have a 12V supply available from a secondary ATX power supply; see Chapter 45, Power Supplies for Feedback for more on that.

Follow the generic wiring plan in Chapter 47, Feedback Device Wiring:

- If your beacon has a motor, be sure to connect a diode between the (+) and (-) leads of the beacon, with the striped side of the diode connected to the (+) lead of the beacon, as described in Chapter 53, Coil Diodes. This is required if there's any sort of motor or coil in the device. Any beacon with a rotating light has a motor inside to drive the rotation.
- Connect +12V from your power supply (the yellow wire for an ATX power supply) to the (+) terminal of the beacon
- Connect the (-) terminal of the beacon to an available output port on your output controller

The beacon might not have specific (+) and (-) terminals. If not, the order of the connections probably doesn't matter.

Electrical interference: If you're using a mechanical beacon with a motor that drives the rotating light, be sure to use a diode as mentioned above. If you still have problems with electrical interference when it runs (for example, USB devices randomly disconnect, or you see random keyboard input on the PC), you might need to add more filtering. The two-inductor filter described for the shaker should work here. See "Electrical Interference" in Chapter 61, Shaker motors for the wiring and parts details.

Check the power: You should check your beacon's power usage to make sure that your output controller can handle the load. The beacons I use draw about 500mA each, so the pair of them requires about 1A. If your product's documentation doesn't specify the power draw, you can test it yourself using an ammeter.

For Pinscape expansion boards: It should be safe to connect any beacon to any MOSFET Power Board output. Those outputs can handle up to about 4A. Nearly any beacon should be well under 1A. It's best to check (measuring with an ammeter if necessary) if you're not sure, though.

If you're using an LedWiz: It's safe to connect a beacon directly to an LedWiz output if the device draws **less than 500mA**. If it's above 500mA, you'll need some sort of "booster". See "Power Limits" in Chapter 50, LedWiz Setup.

DOF setup: In the DOF Config Tool, go to your Port Assignments page. Find the output controller port number where you wired the beacon. Assign this to "Beacon".

Increasing the brightness: Most of the rotating light beacons you'll find come with small incandescent bulbs. If your cab includes flashers and strobes, the beacons might look dim and unimpressive in comparison. This isn't really the beacons' fault; it's more that the LED devices we use for flashers and strobes are extremely bright. They'll put a little incandescent bulb to shame (especially with the darkly tinted plastic dome covering it).

You can increase the brightness somewhat by installing an LED bulb in place of the incandescent that your beacon comes with. Most of the beacons use common automotive bulb types for which LED replacements are readily available. Almost all automotive light bulbs run on 12VDC, so the main trick is to figure out the type of socket base that your beacon uses, so that you can find a bulb that fits. Check your beacon's instructions, or just look at the bulb itself. The bulb type is often printed or embossed somewhere on the metal base of the bulb.

The Peterson model 771 beacons use bulbs with socket type 1156, also known as BA15S. Any LED replacement bulb with the same base and a 12V supply voltage should work.

In case you were wondering, you **don't** have to add extra resistors if you switch to LED bulbs. If you've read the chapter on Chapter 52, LED Resistors, you know that "bare" LEDs require resistors in the circuit to limit the power going through them. But we're not talking about bare LEDs here; we're talking about LED replacements for incandescent bulbs. Any LED bulb that's designed as a plug-in replacement for an incandescent bulb has the necessary resistors built in, so no external resistors are needed.

If you install an LED, and the motor runs but the LED won't light up, you might need to reverse the polarity of the power supply connection to the beacon. Incandescent bulbs aren't polarized, but LEDs are. If the LED bulb won't light, try swapping the (+) and (-) leads connected to the beacon. This might also have the effect of reversing the direction of the motor, but it should make the LED light up. **Warning:** if you've attached a diode to the beacon (required if your beacon has a motor), be sure to reverse the diode at the same time that you reverse the voltage polarity.

Another thing you can do to improve the *apparent* brightness of the beacons is to supplement the beacons with some additional red and/or blue strobe lights. The "22-LED strobes" recommended in Chapter 56, Flashers and Strobes also come in red and blue varieties. So:

- Buy a 22-LED blue or red strobe, matching the color(s) of your beacon(s)
- Mount it on top of your cab next to or behind the beacon, facing the ceiling
- Unlike the advice for strobes in Chapter 56, Flashers and Strobes, **do** use the little control box that comes with the strobe panel, so that the LEDs flash when the beacon is activated rather than turning on continuously.
- Wire the control box to the same output port as the beacon, so that the LEDs start flashing whenever the beacon is running
- Make sure that the added power of the LED panel doesn't overload your output controller port. If the combined load would be too great, use a second port for the strobe panel instead, assigning it in the DOF configuration to the same "Beacon" device.

The added LED panel obviously won't increase the actual brightness of your beacon, but it will increase the overall amount of flashing red or blue light during a beacon display, which will create the impression of a brighter beacon. I recommend this approach because I use something like it myself to good effect. (My setup is a little more custom, but it's basically the same idea. I use some extra LEDs around the perimeter of my custom fan housing instead of the 22-LED panels, and I use a custom-programmed flashing pattern controlled by a Teensy microcontroller.)

58. Undercab Lighting

Some pin cab builders put LED lighting on the underside of their cabinet, to create a pool of light around the machine. This can also be used on the back of the main cabinet and backbox, to create an aura effect on the wall behind the machine. Unlike most of the other feedback equipment, this isn't meant to replicate a feature of the real machines (not an original feature, anyway, although undercab lighting is a popular after-market mod). This is more of an extra feature to add to the arcade atmosphere.

Undercab or behind-the-cab lighting is usually implemented with LED light strips. These are narrow plastic strips with small LEDs mounted about every half inch. The strips can be cut to any length, so they can be adapted to cover whatever area you want to cover. The closely spaced LEDs provide a nice diffuse glow over the whole area.

LED strips are easy to find on eBay, Amazon, etc., because a lot of people use them for home lighting applications.

You can buy strips with single-color LEDs, but for our purposes, you'll want the full-color "RGB" type. These combine red, green, and blue lighting elements that can be controlled independently, allowing them to display any color by adjusting the mix of the three color components. That lets DOF show different color effects for each game, and optionally change the colors in sync with the game action during play.

There are two rather different kinds of the RGB strips that look very similar on the surface, but serve different functions: the standard "dumb" type, and the "addressable" type. The standard type lets you set all of the lights on the strip to the same color. You can set them to any color, but all of the LEDs will show that same color. The addressable kind lets you set each individual LED on the strip to its own separate color.

For undercab lighting, most people use the standard type, because the goal is simply to create a diffuse pool of light. The addressables are much more expensive and much more complex to set up, which makes them hard to justify for ambient lighting. That's not to say you shouldn't use addressables at all - they actually have other uses apart from undercab lighting where you'd want them; see Chapter 65, Addressable Light Strips. But for undercab lighting, the simpler "dumb" type is more appropriate.

Parts

Light strips: eBay usually has the best prices on these light strips. Search for **5050 RGB LED strip**. Matching listings typically use photos that look something like the one at right.

The strips on eBay are most commonly sold in 5-meter reels (labeled "5M" on eBay). 5 meters is about 16 feet; for a full perimeter around the underside of your cabinet, you'll need about 3.5 meters, so one 5-meter reel will do the job. Don't worry about finding the exact length you need, as the strips can be cut to any length.



Don't buy a "kit"; look for a bare reel only. Many eBay sellers offer kits that come with accessories such as remote controls or wall-wart power adapters. You won't

need those in a pin cab installation.

The eBay seller pages for the reel strips often combine a bunch of different options. Here are the common choices and what you should select:

- 3528/5050/5630 type: choose 5050 (usually the only type available in RGB)
- Color: RGB, sometimes called multi-color or full-color
- Length: 5 meters (5M)
- Waterproof/non-waterproof: choose the non-waterproof
- Number of LEDs: 60 per meter (the seller might indicate this as "300 LEDs" if it's a 5-meter strip)

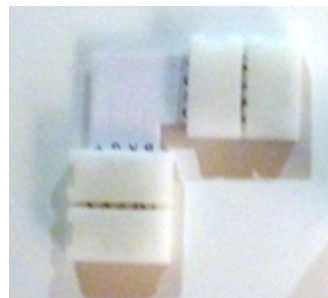
Wire connectors: You can find solderless wire connectors for the strips like the one pictured below. Search for "4-wire RGB light strip connector". Look for the type with the white plastic connector at one end. The white plastic bit is designed to clamp onto the end of a light strip to connect it without soldering.



These are optional. You can also just solder wires directly to the copper pads on the strip. That's a little tricky, though; the solderless connectors are easier.

Note that the black plastic plug on these connectors mates with the LED amplifiers below, so these are also helpful if you're going to be using any of the amplifiers.

Corner connectors: If you're going to set up a ring around your cab's floor, corner connectors can be very helpful. These look like the ones pictured at right.



Amplifiers: If you're going to attach the strips to an LedWiz, PacLed, or standalone KL25Z, you **must** use an amplifier to power an undercab strip. These aren't necessary if you're using a Pinscape power board to control the strip, as long as your combined strips aren't more than about 4 meters long. (And if you do need more than 4 meters of strips, you could still avoid the amps by breaking the strips up into 4 meter sections, and attaching each section to its own separate set of power board output ports.)

You can find amplifiers built specifically to work with these strips on eBay, by searching for "LED amplifier". Look for the "mini" amplifier like those shown at right. Don't buy a "remote control" or color changer device - those can look quite similar, but they'll have buttons to change modes. You just want a plain amplifier, since DOF is going to do all of the color control.



To determine the amount of current the strip will draw, check the specs on the product you bought to determine the Amps per meter it uses, and multiply that by the combined length of your undercab strip(s). The typical power spec is 1.2A per meter per color channel (1.2A per meter for the Red channel, and the same for

Green and Blue). A typical undercab lighting setup needs about 3 to 3.5 meters of the strips, which comes to about 4A per channel. That's way above the LedWiz's 500mA (0.5A) limit, so you'll need to use an amplifier with the LedWiz. 4A is fine for the Pinscape power boards, though.

Positioning

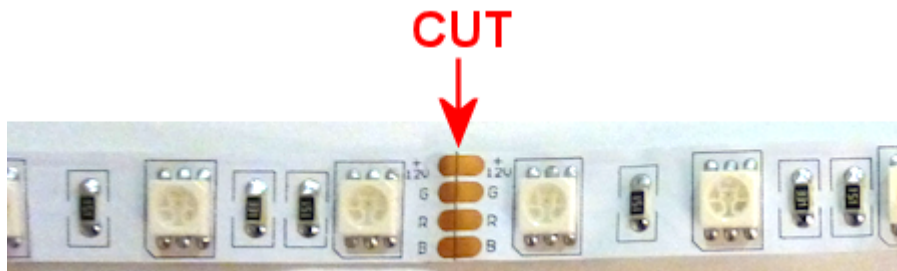
The arrangement I recommend is to make a ring around the perimeter of the cabinet, close to each outside edge. I'd attach the LEDs to the flat bottom panel, not to the edges of the side walls, as the glue will adhere better to the flat plywood surface than to the more porous edges.

Another place that some people like to put the strips is on the back of the backbox. A couple of vertical strips, one at each edge, should work nicely there.

Installation

The first step is to measure out the length you need for each run, and cut the strip into the required lengths.

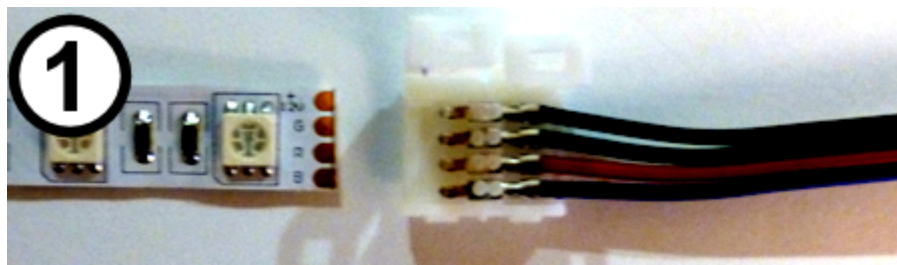
You can cut the strip only at certain points, where the copper pads are exposed. On all of the strips I've seen, the cut points are spaced about 2 inches apart, every 3 LEDs. There should be a line marked on the strip down the middle of the copper pads. Use a pair of scissors to cut right down the center line.



When deciding where to cut, take into account the extra space you'll need for the wire connections to the end of the strip and any corner connectors you'll be using.

Connecting wires to the strips: The easiest way to wire the strips is to use the solderless connectors mentioned above.

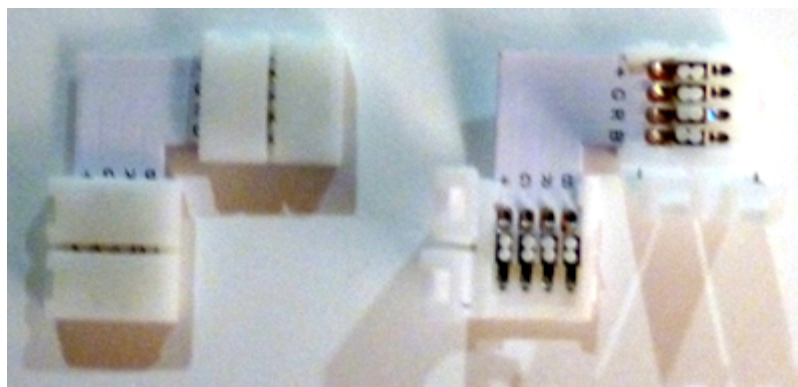
To use these, pry open the white clamp cover (there's a little locking tab on one side), so that it looks like image (1) below. Position the end of the strip so that the copper pads on the strip fit over the metal prongs in the connector, as in (2) below. Note the labeling on the strip: the pad labeled "+12V" should line up with the black wire, "G" should line up with the green wire, "R" should line up with the red wire, and "B" should line up with the blue wire. Once the strip is positioned, close the white plastic cover and make sure the lock tab engages. This should hold the strip in place mechanically and form a good electrical connection to the wires.





If you prefer, you can solder your own hookup wire directly to the copper pads on the strips. I've found it tricky to get the solder to stick, though, since the pads are so small. And even when you get a wire to stick, the solder joints will be fragile, due to the flexibility of the plastic strip. If you do decide to use solder, test your work by applying power to all channels and verifying that the strips light up white, then securely wrap the solder joint and the adjacent inch or so of wire with electrical tape. You want to make the area around the connection stiff enough that it won't bend much while you're working with the strip. The solder isn't nearly as flexible as the plastic strip, so the solder joints will break easily if you flex the plastic even slightly.

Connecting corners: If you're making a ring around the perimeter of your cab, use the corner connectors mentioned above to connect adjacent strips at the corners. The corner connectors work just like the solderless wire connectors: snap open the cover of the white plastic bit, fit the end of the strip into it so that the copper pads on the strip line up with the metal prongs in the connector, and snap the plastic cover shut to lock the strip in place.



Corner connectors, with the covers closed (left) and open (right). Fit the end of a strip into each connector so that the copper pads on the strip line up with the metal prongs in the connector.

Attaching to the cabinet: The strips come with a self-adhesive backing, so in principle you can just peel off the backing and stick the strips to the bottom of your cab.

In principle, anyway. In practice, the glue on the backing isn't all that strong, at least when used with bare plywood. On my own cab, the strips seemed to attach well

initially, but they started peeling and drooping after a couple of months. I've heard the same story from several other people. So you might need to shore up the attachment with something stronger: staples, hot glue, epoxy, or a strong double-sided adhesive foam tape.

If you use staples, be really careful not to pound them in so hard that you break the delicate copper traces on the plastic strips. Also be careful that they don't come into contact with any of the exposed metal traces or pads on the strips, as that could cause a short circuit that could damage the power supply.

If you use hot glue or epoxy, put a tarp down while you're working so that you don't drip any glue on the floor.

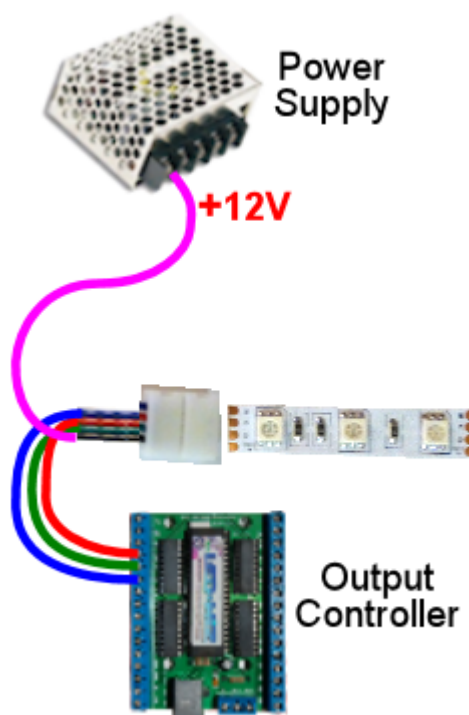
If you're using the wiring connectors and/or corner connectors, it's worth gluing or otherwise fastening those to the cab as well, as they add some weight that would otherwise be borne by the adjacent strips.

Wiring

Direct connection: If you're **not** using an LED amplifier (e.g., you're connecting the strip directly to a Pinscape power board), the wiring follows the standard plan for typical RGB devices, as illustrated below.

Warning! Don't use a direct connection like this with an LedWiz or standalone KL25Z. Doing so will damage or destroy the controller. The current (amperage) is too high for either one to control directly, and the voltage is too high for the KL25Z. You also can't use direct connections with a PacLed, since those are also limited to low current levels. Skip down to the part about using LED strip amplifiers below.

You **can** use a direct connection like this with the Pinscape expansion boards, using Power Board outputs.

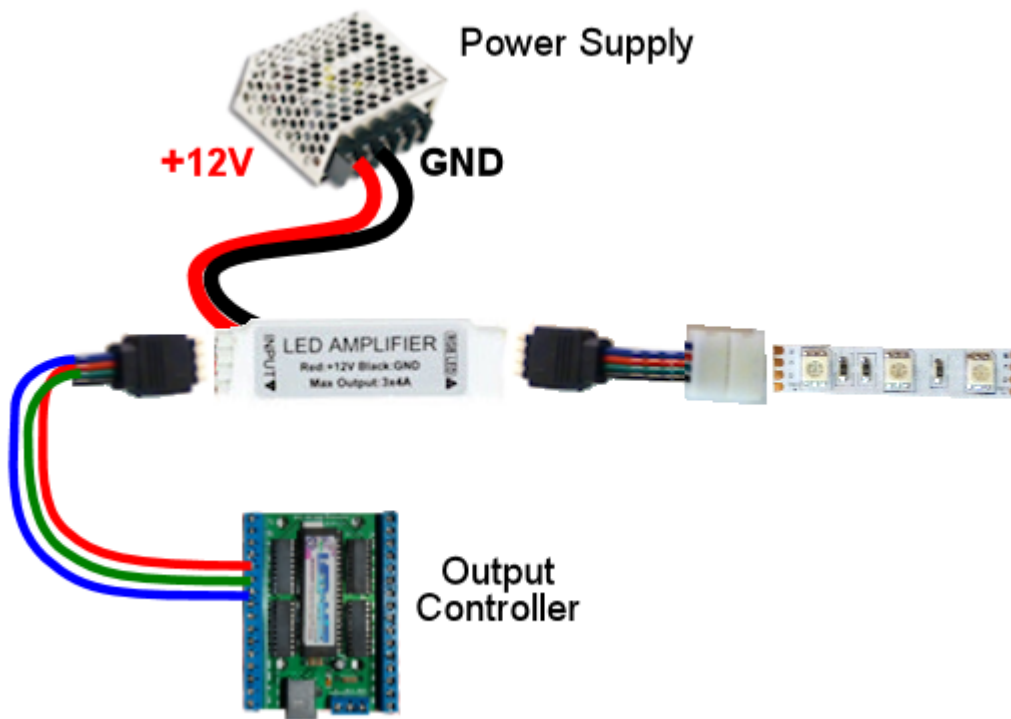


- Attach a connector to the end of the LED strip as described above.
- Cut off the black plug from the connector.
- If necessary, solder an additional length of ordinary hookup wire (22 to 24

gauge should work) to each of the RED, GREEN, and BLUE wires coming out of the connector, to make the wires long enough to reach your output controller. Also solder an additional length of hookup wire to the black wire from the connector, but in this case you need it to reach the +12V terminal on your power supply.

- Connect the red, green, and blue wires to available ports on your output controller.
- You must follow the standard DOF rule about RGB devices: the red, green, and blue wires must be connected to consecutively numbered ports, in red-green-blue order. For example, if you plug the red wire into port 15, green has to go to port 16, and blue has to connect to port 17.
- Connect the black wire from the LED strip connector to the +12V terminal on your power supply. You can use the secondary ATX power supply for this; see Chapter 45, Power Supplies for Feedback.

Connecting through LED strip amplifiers: The wiring with an amplifier involved looks like this:



- Attach a connector to the end of the LED strip as described above.
- Plug the black end of the connector into the "Output" side of the LED amplifier. The four-pin black plug on the solderless connector mates with the LED amplifier's Output socket. To orient the plug properly, match the **black wire** on the connector to the arrow or "+12V" printed on the LED amplifier body. This should match up with the "+12V" marking on the LED strip itself.
- Plug another connector of the same into the "Input" side of the LED amplifier. Again, the black 4-pin plug on the connector should mate with the Input socket on the amp. Match the black wire on the connector to the arrow or "+12V" marking on the amp body.
- Cut off the white plug at the other end of the connector you just plugged into the Input side of the amp.
- The black wire from this bundle can be left unconnected.
- If necessary, solder an additional length of ordinary hookup wire (22 to 24

gauge should work) to each of the RED, GREEN, and BLUE wires coming out of the connector, to make the wires long enough to reach your output controller.

- Connect the red, green, and blue wires to available ports on your output controller.
- You must follow the standard DOF rule about RGB devices: the red, green, and blue wires must be connected to consecutively numbered ports, in red-green-blue order. For example, if you plug the red wire into port 15, green has to go to port 16, and blue has to connect to port 17.
- The LED amplifier has two loose wires hanging out of the back, one red and one black. These are the power connections for the amplifier. They simply connect directly to your 12V power supply. Connect red to +12V and black to ground. You can use the secondary ATX power supply for this; see Chapter 45, Power Supplies for Feedback.

DOF setup

In the DOF Config Tool, go to your Port Assignments page. Find the numbered port that you wired to the RED channel of the light strip. Set this to either "RGB Undercab Smart" or "RGB Undercab Complex". In standard DOF fashion for RGB devices, this will set the next two ports in numerical order to the Green and Blue color channels for the same device. (This is why you have to wire the red, green, and blue wires to consecutively numbered ports in red-green-blue order. It's just the way the Config Tool works with RGB outputs.)

"Smart" and "Complex" are both for undercab lighting, so you can choose either one. They have slightly different behavior to accommodate different taste:

- RGB Undercab Smart shows a fixed color that's customized for each game. The color generally reflects the predominant color of the cabinet artwork for the original arcade version of each game.
- RGB Undercab Complex uses the same color as RGB Undercab Smart as the base color, but also animates the color in response to game action. For example, it might flash white when the strobes fire, or show the same color as the flasher lights in response to some events.

The choice is just a matter of taste. Some people like the added lighting effects of the "Complex" option, and others find it too distracting. If you're not sure which one you'd prefer, I'd recommend giving the Complex option a try first; you can always switch to the more subdued Smart option later if the Complex effects are too much.

59. Flippers, Bumpers, and Slingshots

Real pinball machines are mechanical contraptions with lots of moving parts. It's what makes them stand apart from just about every other kind of arcade game. When you translate pinball into a video game, you lose that physicality. In a virtual pin cab, you can bring back some of the mechanical feel of the real thing by adding similar moving parts that can be actuated in sync with the game action.

The moving parts on a real machine that produce most of the tactile experience for the player are the ones controlled by solenoids: flippers, slingshots, bumpers, kickers. All of these are driven by powerful magnetic coils that propel the metal parts at high speed. The kick of the solenoids is palpable while playing. This chapter is about reproducing that tactile solenoid effect in a virtual cab.

How many? Where do they go?

It's up to you which devices to include in your setup. The common setups, which DOF supports most readily, use two devices to simulate flippers, two to simulate slingshots, and four to six devices to simulate pop bumpers.

Flippers: The most basic solenoid setup uses just two solenoids, one for each flipper. These should be positioned near the location where the actual flippers appear in the video presentation, near the bottom of the TV. You can mount these inset a bit from the sides of the cabinet, where the actual flipper solenoids would be on a real machine, or you can mount them directly to the cabinet walls. If you're using slingshot solenoids in addition to the flippers, you might want to use the inset positioning for the flippers, and position the slings further apart, to make the sounds more noticeably separated in space.

Slingshots: These go just behind the flipper devices. Most people mount these directly to the side walls of the cabinet, since real slingshots are also close to the walls.

Bumpers: There are two common configurations for bumpers. The "deluxe" arrangement uses six devices, arranged as a row of three across roughly the middle of the playfield TV, and a second row of three closer to the rear of the TV. This provides devices at enough different places that DOF can do a good job of placing each sound effect close to its simulated origin in the game. The other common arrangement uses a single row of three across the middle of the TV, and one more near the rear center.

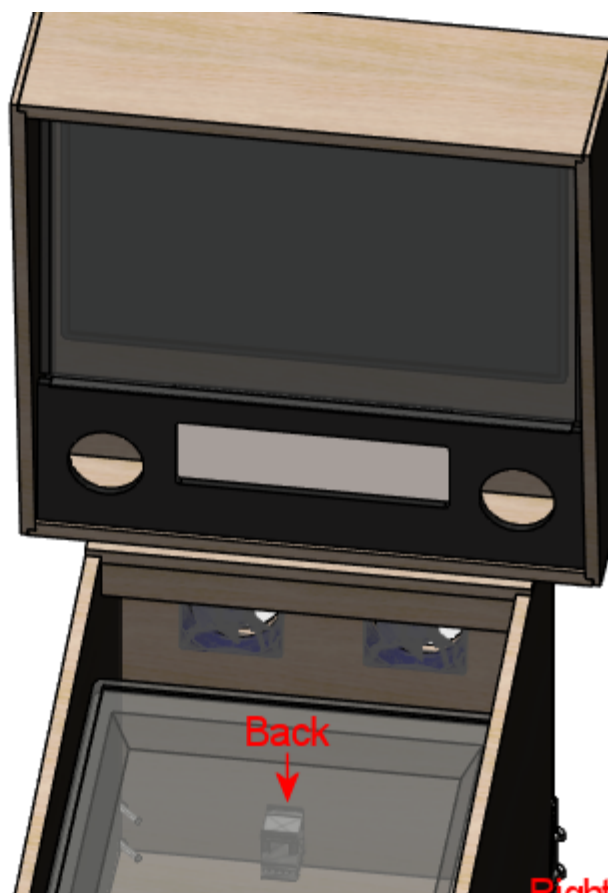
DOF refers to these two common bumper configurations as "10-Bumper" and "8-Bumper" setups. That's a bit confusing, because it doesn't really mean *bumpers*. In both cases, two of the devices are flippers and two are slingshots. So what DOF really means is "10-Solenoid" and "8-Solenoid".

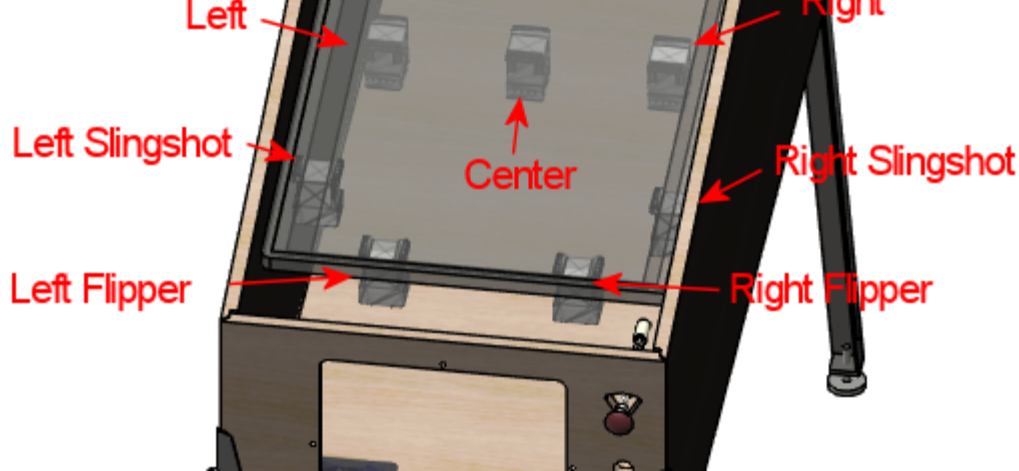
Here's what the two arrangements typically look like in a cab.





"10-Bumper" placement. The actual "bumpers" are the six devices across the middle and back playfield area. The two devices at the front are the flippers, and the two just behind those are the slingshots.





"8-Bumper" placement. The actual "bumpers" are the four devices across the middle and back playfield area. The flipper and slingshot devices are the same as in the "10-Bumper" layout.

Parts

The most effective way to reproduce the sensory effects of a pinball solenoid is to use some of kind of solenoid.

In a real pinball machine, they have to use a specific mechanism for each job, because each mechanism has to do the actual work of propelling the ball (or whatever it's meant to do). In a virtual cab, we don't have that constraint. Our devices don't have to actually do anything physical in the game; they just have to produce the right sensory impression. So we're free to substitute anything that sounds and feels similar. That opens up our options and lets us consider other properties when we're selecting devices, such as cost, size, durability, and power needs.

If you open up the field to consider anything with a solenoid in it, you have an overwhelming number of options. Fortunately, people have been building pin cabs for long enough now that a few winning options have emerged, so you won't have to spend months doing research on this; you can just pick one of the common solutions below. Like most pin cab features, everything has some tradeoffs, so there's not a perfect one-size-fits-all solution. We'll give you the pros and cons of each to help you decide.

Note that you don't have to settle on a single device type! Many pin cab builders do use one device type for everything, but others think it's a plus to use different devices for different roles, so that each type of device sounds a little different. For example, you might use the Siemens contactors for the flippers, and starter relays for the bumpers and slingshots.

Contactors

A contactor is a type of electrically controlled switch used in high-power applications, such as controlling industrial equipment or large appliances like air conditioners. "Contactor" is just another word for "relay" - they're fundamentally the same thing - but they call it a contactor when it's the higher power type.

On the inside, a contactor is a mechanical switch controlled by a solenoid. The solenoid moves a set of mechanical switch blades when energized. Because of the high power that a contactor is designed to handle, the switch blades have to be hefty, and they have to move across a wide gap when switching.

It's that big moving switch element that makes contactors useful for our purposes. The large moving parts make a nice "thunk" when the device switches on and off, strong enough to resemble a pinball flipper, slingshot, or bumper.

What to buy: The type of contactor that's popular with pin cab builders is made by Siemens. One particular known quantity that many cab builders use is Siemens model number 3RH1140-1BB40 (pictured at right). It's safest to look for that exact part, but Siemens makes a large number of similar devices, and I suspect that any Siemens model that physically resembles this one will have substantially the same component parts inside and will produce similar sensory effects.



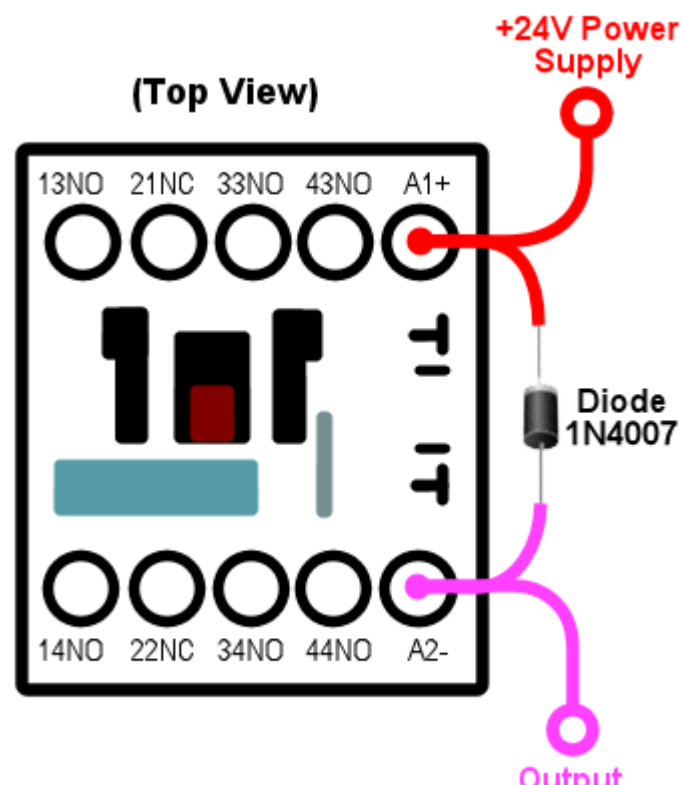
Many other types of contactors are also available from other manufacturers, so there are undoubtedly many more options out there that would work. However, there's no reason to assume that any random contactor would sound "right". If you're considering another device, ask in the forums to see if anyone else has tried it; or try it yourself if you want to be a pioneer, but ideally buy it from someone who will let you return it if it doesn't do the job.

Cost: You *can* buy these devices new, but you shouldn't: they're \$50 and up apiece new. Given the performance tradeoffs, they're not worth the money; if you have that kind of budget, real pinball parts would be a far better investment. So you should instead buy them used on eBay, where they can be found for about \$10 apiece. You might have to be patient to find them at a reasonable price, since they seem to be a fairly low-volume item on eBay.

Power supply: The Siemens contactors require 24VDC power. That's not available from a computer ATX power supply, so you'll need a separate power dedicated 24V supply. See Chapter 45, Power Supplies for Feedback for advice.

The Siemens contactors draw about 400mA each. I'd plan on sufficient power for three or four of them to fire simultaneously, so be sure your 24V power supply is rated for at least 2A (equivalent to 48W).

Wiring: Here's how to hook up the typical Siemens contactors:



Note that most of the terminals on the contactor are left unconnected. The other terminals are all intended to be wired to whatever circuits the contactor is controlling, but in our case, we don't control anything at all with the contactor. It's just there for the sound effect it makes when it switches on and off, so we only connect the terminals that control its coil, typically A1 and A2 on the Siemens devices.

The diode is required to protect the output controller and computer electronics from noise from the magnetic coil inside the contactor. See Chapter 53, Coil Diodes. The stripe on the diode must connect to the positive (+) power side.

Pros: The biggest "pro" with contactors is that they're happy to be left in the "on" position indefinitely. That's not at all true of most solenoids: most solenoids will overheat if left on for long periods, and many can only tolerate a couple of seconds at a time. For simulating flippers, the ability to stay on for long periods is critical, because the player could hold the flipper button indefinitely to trap a ball.

The tonal quality of audio/tactile effect of the Siemens contactors is another plus. It's a nice solid mechanical "thunk" that creates the impression of a large device.

Other good features: they're compact, self-contained, easy to install, run on fairly low DC voltages (the Siemens devices referenced above run on 24VDC), and use relatively little power (about 350mA for the common Siemens device).

Cons: The biggest negative is that the effect is far weaker than a real pinball solenoid (even though the tonal quality is good). Also, they require 24V, which means you have to add an extra power supply for them. The Siemens devices are pricey, unless you buy them used, and used ones sometimes have a lot of wear that greatly reduces the strength of the "thunk" effect, which is the whole point for our purposes. Used ones might not be entirely reliable, either; a couple of my own Siemens contactors are intermittently "sticky", where they'll stuck in the on or off position for a while and won't make any noise. I've heard from other people seeing the same problem.

Recommendation: The Siemens contactors are a popular way to implement the simulated flippers and other solenoids because they're simple to install and they won't overheat if left activated for long intervals. The tolerance for long activation times makes them especially attractive for flipper effects. They're the best bet if you want to keep things simple. But the effect isn't nearly as strong as the real thing, so if realism is a high priority, look at other options.

Automotive starter solenoid relays

This is another kind of high-power relay, in this case specific to cars. Starter solenoids are used in automotive engine starting systems; they switch the high-current connection between the battery and the big solenoid that cranks the engine when you turn the ignition key. As such, they have very much the same purpose as contactors - switching high-power loads through a relay switch - and similar construction.

Like contactors, these have big moving parts inside that create a nice "thunk" effect when actuated. The effect is similar in tonal quality to the Siemens contactors, and it's often quite a bit stronger, but usually not as strong as real pinball coils.

Starter relays are cheaper and easier to find than the Siemens contactors. Many types sell for under \$10 at auto supply stores, eBay, Amazon, Wal Mart, and other

big retailers. Like the contactors, they're self-contained and easy to install.

Unlike contactors, these devices will overheat if energized for extended periods, so they're not ideal for flippers. You might be able to use them for flippers if you take special precautions, such as using Flipper Logic (see below). Overheating is less of a risk when you use these devices for bumpers and slingshots, since those typically only fire in brief bursts in normal play.

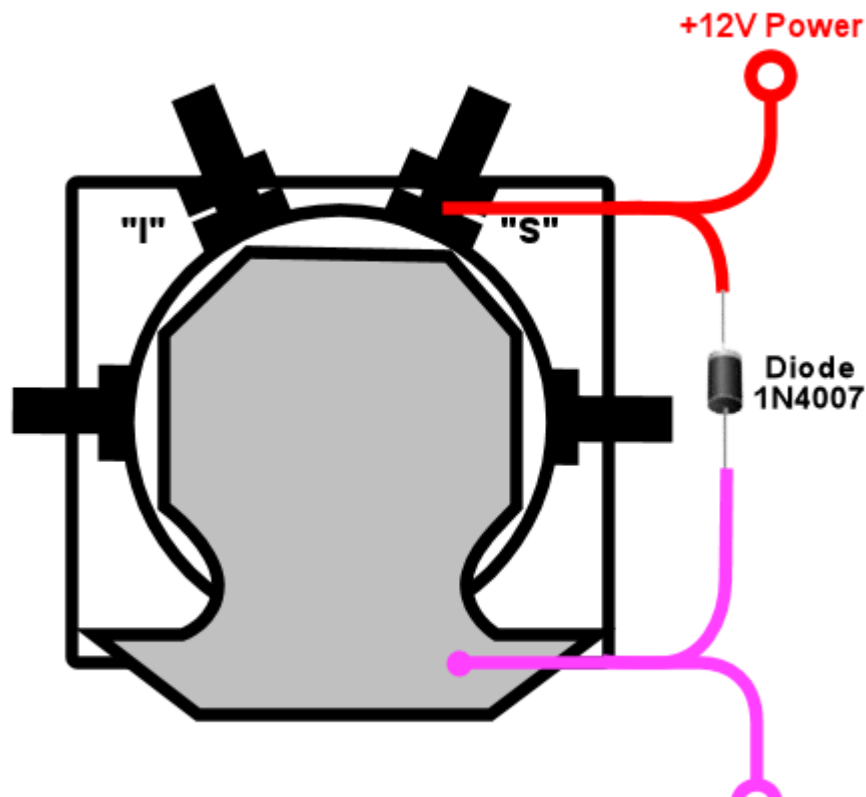
Reliability is also a concern. If you think about it, these devices aren't typically fired all that often in their intended application of starting a car. A pin cab will subject them to much heavier use than the manufacturers presumably expect (and design for). Some people on the forums have reported that they need to replace these as frequently as every few months, which I'd find overly frustrating. But I'm sure the durability varies a lot according to how you use your cab and which specific devices you buy.

What to buy: The type most often mentioned on the forums is the Ford SW3 type (pictured at right). As with the contactors, numerous other options are available, but this one is a known quantity that other pin cab builders have found to work well.



Power supply: Almost every automotive device runs on 12V DC (but as always, check the product documentation or seller page to confirm details for the actual product you buy). You can use the 12V output from your secondary ATX power supply. Note that these devices place a large load on the power supply - about 6A, or 72W - so make sure you have a powerful enough 12V supply, and don't connect them to the same ATX power supply running your computer motherboard. When calculating power needs, take into account that there will be times during normal play when several of the devices fire simultaneously, so multiply the amps/wattage rating by three or four to figure how much total power you might need. See Chapter 45, Power Supplies for Feedback.

Wiring: For the Ford SW3 type commonly used, the wiring plan looks like this:



On the Ford SW3 type, you'll find four screw terminals sticking out from the body of the device. The ones on top (in the orientation shown above, anyway) should be labeled "S" and "I". The ones sticking out from the sides aren't typically labeled. In addition, the large metal plate that looks like a mounting bracket is indeed a mounting bracket, but it's also the electrical ground connection for the device. The terminals of interest for pin cab use are the one labeled "S" and the mounting bracket/ground plate.

If your solenoid isn't labeled like the Ford type above, you'll have to figure out which terminals are which. On any automotive device, you can count on any exposed metal on the outside being wired to the electrical ground, so look for an unpainted mounting plate or a socket for a mounting bolt. Now get out your multimeter. Set it to measure resistance. Connect one probe from the meter to the ground point (that exposed metal or mounting plate or whatever), and use the other probe to measure the resistance to each of the other terminals, one at a time. One terminal should have a low but non-zero resistance, probably something like 3 or 4 ohms. That should be the +12V coil terminal - the one that corresponds to the "S" terminal on the Ford devices.

Connect +12V from the secondary ATX power supply to the "S" terminal. Wire the mounting bracket to an available port on your output controller.

The diode is required to protect the output controller and computer electronics from noise from the magnetic coil inside the contactor. See Chapter 53, Coil Diodes. The stripe on the diode must connect to the positive (+) power side.

These are high-power devices. Do not connect them directly to an LedWiz output; you'll need some kind of booster circuit or relay with an LedWiz. If you're using Pinscape expansion boards, you can connect these directly to any power board port or chime board port.

Pros: Inexpensive, readily available, self-contained, easy to install, run on 12V DC power. The sound effect they produce has a good tonal quality, and it's stronger than the Siemens contactors (though probably weaker than real pinball solenoids).

Cons: They're not designed for continuous activation, so they shouldn't be used for flippers without special precautions. Some types might not be durable enough to last more than a few months with the heavy use they get in a pin cab. For most starter relays, the effect isn't as powerful as real pinball solenoids.

Recommendation: These are a good choice for everything except flippers, and they can even be used for flippers if use something like the Pinscape "Flipper Logic" feature to reduce hold power. Many people think they sound more realistic than the Siemens contactors. Durability is the main drawback; some people on the forums have had to replace them after only a few months. Given their low cost, though, that might be an acceptable risk, as long as your cab allows easy maintenance access.

Real pinball mechanisms

If you want maximum realism, you can use real pinball assemblies for flippers, slingshots, and bumpers.

This is rarely done, but not unheard of. A few cab builders on the forums have reported going this route. It has a number of challenges compared to the other methods:

- It's really expensive. The relevant pinball assemblies run about \$50 each, so a full set (two flippers, two slings, six bumpers) comes to about \$500.
- The assemblies take up a lot of room in the cabinet.
- They require a high-voltage (50V), high power (600W) power supply. That's expensive, and many pin cab builders are uncomfortable working with such hazardous voltages.

What to buy: Buy these at any pinball supplier (Pinball Life, Marco Specialties). You can also buy them used on eBay, but my standard warning about used pinball parts on eBay applies: they'll be beat up and the sellers all think you're an idiot who wants to pay new prices for old parts. At least price the new ones first so you know whether an eBay bargain is really a bargain.

The key word when searching the pinball vendors is "assembly":

- "Flipper assembly" (e.g., Williams reference C-13174-L, C-13174-R)
- "Slingshot assembly"
- "Bumper assembly" (e.g., Williams reference A-9415-2, B-9414)

Important note on flippers: Real flipper assemblies from the 1980s and 90s should have "dual coil" arrangements. You won't actually see two separate coils - the two coils are wound around a common core and look like a single coil. But you should see three terminal wires coming out of the coil. The flipper assembly should have an end-of-stroke switch that diverts power from the high-power "lift" coil to the low-power "hold" coil when the flipper reaches the end of its arc.

The dual-coil arrangement makes it safe to hold the flipper button on for long periods. The lower-power hold coil is specifically designed for continuous activation without overheating.

I mention all of this because some newer Stern assemblies *don't* use the dual-coil design and aren't safe to use in a virtual pin cab without special software provisions. If you use one of these newer assemblies, it's critical that use something like the Pinscape Flipper Logic feature (see below) to reduce power to the flipper coil when it's held on. Because of this additional complication, I'd recommend avoiding the Stern assemblies and using a traditional dual-coil Williams/Bally assembly. That should be safe to use with any controller without any additional setup work, since the hold power reduction is handled directly in the flipper assembly hardware itself.

Cost: About \$50 per assembly. A full "10-Bumper" setup (in DOF parlance - two flippers, two slings, six bumpers) is about \$500.

Power supply: Real pinball coils in modern machines require 50V DC power supplies. Most of them draw 3-5A. To allow for multiple devices firing at once, you should plan on a minimum 10A (500W) capacity.

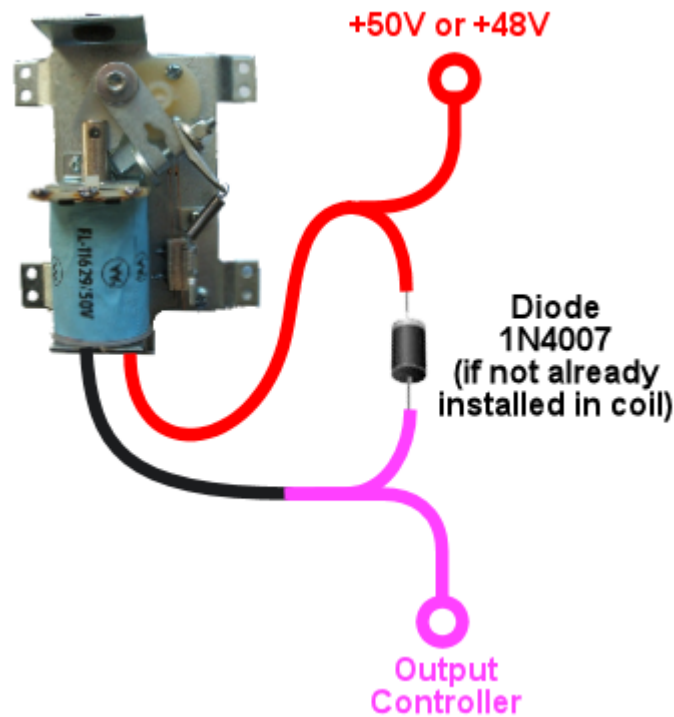
50VDC power supplies are expensive hard to find. It's easier to find a 48V supply, which will work about as well. 48V supplies are widely used for LED lighting applications; you can find them in the 500W range on eBay for about \$50.

Warning! 50V (or 48V) is a hazardous high voltage that can cause electric shock. With most of the pin cab output controllers (including LedWiz and Pinscape), the controller requires you to wire solenoids so that one terminal is connected directly to the "+" power supply voltage. This means that +50V is always present at one terminal of each coil, *even when the coils are off*. Real pinball machines have a hard-wired safety interlock switch in the coin door that cuts off all coil power to the playfield when the coin door is opened, to protect an operator from shock hazard

while working inside the cabinet. If you install real coils and a 50V supply, I'd strongly recommend using a similar safety interlock. The switch should disable the +50V power supply when the door is open.

Wiring: Each coil will have two terminals for the power connections. Connect these following the generic output device wiring plan (see Chapter 47, Feedback Device Wiring).

Many of the flipper, bumper, and kicker assemblies for modern machines include diodes pre-installed on the coils. If a coil is already installed, it should have markings for the "+" and "-" wires. This is often by wire color: red is "+" and black is "-".



If a diode *isn't* already installed, you must add one. See Chapter 53, Coil Diodes. When a diode isn't present, the coil itself will be unpolarized, so the order of the wires doesn't matter. Just be sure that the diode you add is installed with the correct polarity.

If you're using an LedWiz, these devices require some kind of booster or amplifier, because they use much more power than an LedWiz can handle. If you're using Pinscape expansion boards, you can connect these to any Power Board or Chime Board port.

I'd recommend using the Pinscape Chime Boards, if you have them, for the bumpers and slingshots (but **not** for the flippers). Those coils will overheat and melt if they get stuck on by a software fault. If you don't have any Chime Boards but you're using the Pinscape software, use the Flipper Logic feature (see below) to cut off device power if a port gets stuck on, by setting the "hold power" to 0%.

Pros: Not merely realistic - *real*.

Cons: Expensive; take up a lot of space; vulnerable to overheating if they get stuck on; require high-voltage power supply; electric shock hazard.

Recommendations: The ideal solution, if cost is no object and you're willing to work with the high voltage.

Open-frame solenoids

The core element of every solenoid-based pinball device is the solenoid itself, so some cab builders bypass the various devices built *around* solenoids (like the contactors and starter relays mentioned above) and just use standalone solenoids. You can find options on eBay, and from hobby robotics suppliers like Adafruit and Sparkfun.



Plain solenoids don't seem to be widely used on pin cabs, and many of the people who have tried them seem to be dissatisfied with the results, judging by the forum discussions on the subject. I think it's possible to create a good effect with a plain solenoid, but it's much more challenging than with the pre-made devices like contactors and starter relays. There are two main reasons for this.

The first is cost. Many pin cab builders who use plain solenoids do so because they think it's a cheaper way to accomplish these effects. You can find cheap solenoids, but the cheap ones tend to be small toy solenoids that aren't nearly as powerful as pinball coils. The effects they produce will be correspondingly small and unimpressive. They'll just make metallic clicks. If cost is your main concern, you're probably going to do better with something like a starter relay.

The second challenge is that pinball assemblies we're simulating aren't *just* big solenoids. They're big solenoids with other moving parts attached. Typically big, heavy, metal parts. Those other parts are integral to creating the right sensory effect. If you want to produce a convincing effect, you'll have to contrive your own additional moving parts similar to what's in the pinball assemblies. This is what makes the contactors and starter relays so plug-and-play: they have their own built-in moving parts that happen to produce effects similar to what we're after.

I think this can be a promising avenue to pursue, but more so if you're willing to buy somewhat more expensive parts and do some experimentation to find the right combination of attached parts to produce the right effect.

What to buy: On eBay, look for "open frame solenoid". I'd avoid the small 12V devices; those are little toy solenoids that will just sound like metallic clickers. Look for something 24V or above. Sparkfun sells a 36V solenoid that a couple of people on the forums have mentioned favorably.

Cost: \$5 and up. The Sparkfun 36V device sells for about \$20 as of this writing.

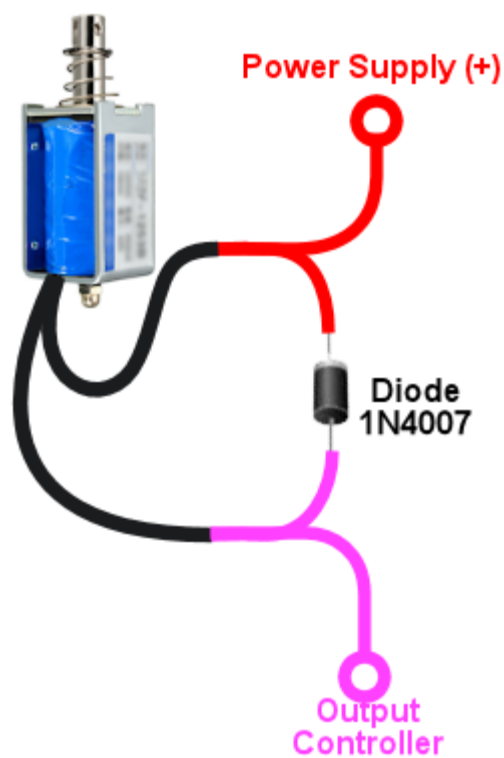
Power supply: Varies by device. Check the specs on what you buy to determine what voltage you need.

If you have to buy an additional power supply, pay attention to the Volts and Amps required by the device. Make sure the power supply has a voltage that is the same as the solenoids, and that it provides **at least** the required amperage. If you're using multiple devices of the same type, the power supply will need to provide enough power for several of the devices simultaneously; I'd multiply the solenoid amperage by 3x or 4x to get a suitable minimum.

If you see specs in Watts for either the solenoid or power supply, you can convert between Watts and Amps using this formula:

$$\text{Watts} = \text{Volts} \times \text{Amps}$$

Wiring: The solenoid should have two wires attached. Solenoids aren't polarized, so the order of attaching the wires doesn't matter. Connect one lead to the (+) supply voltage (use the correct voltage for the solenoid), and connect the other to your output controller, as shown below.



The diode is required to protect the output controller and computer electronics from noise from the magnetic coil inside the contactor. See Chapter 53, Coil Diodes. The stripe on the diode must connect to the positive (+) power side.

Pros: Many options are available; infinite customization possibilities.

Cons: Sensory effects are less predictable than with contactors or starter relays; you might have to experiment with several devices and several mounting styles to get a satisfactory effect. More complex to set up than contactors or starter relays if you want to contrive additional moving parts to improve the sound effect. Smaller devices will produce weak, tinny effects. Most solenoids will overheat if actuated for more than a couple of seconds at a time, so special measures (such as Flipper Logic) might be needed, especially when used for flippers.

Recommendations: If you're thinking of using cheap eBay solenoids to save money, I'd reconsider: you'll probably be unhappy with the results and end up replacing them with something better, so your total cost will be higher than if you just started with something better. But plain solenoids might work for you if you're willing to buy more expensive ones with more force, and you're willing to do some extra work to improvise additional moving parts to simulate the action of the full pinball assemblies.

DOF setup

In the DOF Config Tool, go to your Port Assignments page. Find the numbered ports where you wired your solenoid-type devices. Assign each one to the appropriate output device.

If you're using a full 10-device setup (see the diagrams above):

- Flipper Left/Right
- 10 Bumper Middle Left/Center/Right or Back Left/Center/Right
- Slingshot Left/Right

If you're using an 8-device setup:

- Flipper Left/Right
- 8 Bumper Left/Center/Right/Back (if you're using an 8-device setup)
- Slingshot Left/Right

"Flipper Logic"


If you're using a Pinscape controller, a special feature is available to help avoid overheating devices that aren't designed to be activated for long periods. This isn't necessary with contactors or most real flipper assemblies, since those are specifically designed to tolerate continuous activation. It's useful for almost everything else: all other standard pinball coils, automotive starter relays, and miscellaneous open-frame solenoids.

The idea behind Flipper Logic is to simulate the way real pinball machines solve the overheating problem for their flipper coils. Real flipper coils would have the same overheating issue as our substitutes if they didn't take special measures. On older pinball machines, the special measure is that they use two separate coils in the flipper. A high power "lift" coil provides the initial jolt of power to flip the flipper and propel the ball, and a low-power "hold" coil takes over as soon as the flipper is all the way up. (This isn't obvious looking at them, because the two coils are wound into a single body. But electrically, there are two coils there.) In the real machines, an end-of-stroke switch in the flipper mechanism redirects power from the lift coil to the hold coil.

In some newer Stern machines, they keep the hardware simpler by using a single high-power coil, and solve the heating problem through software, by modulating the power using PWM controls. When you press the flipper button, the software initially applies full power to the flipper (simulating the "lift coil" power). If you continue to hold the button for more than a brief time, the software automatically reduces the power through PWM (simulating the lower "hold coil" power). The lower power is enough to keep the flipper flipped, but it's low enough that it won't overheat the coil even if it's held on for a long time.

Flipper Logic works like the newer Stern machines, reducing the power to the port after a brief period at full power. This makes it possible to use more types of devices as a flipper substitutes, by avoiding the overheating problem common to many solenoids.

You can activate this feature on a port-by-port basis, for any PWM-capable port. To activate it:

- Run the Pinscape Config Tool
- Go to the Settings page for your controller
- Scroll down to the Output Ports section
- Click the flipper icon () at the right side of the port you want to set up

This lets you enter two parameters: Initial Time and Hold Power. The Initial Time specifies how long the port receives full power each time it's activated. You can adjust this from 50ms (1/20 of a second) to 800ms (0.8 seconds) in 50 millisecond increments. The Hold Power is the PWM power level that's provided to the port after that point, as long as the port is held on. This can be adjusted from 0% to 100% in increments of about 7%.

To determine the proper reduced power level for a device, you'll have to experiment with it. Each device is different. Start at the lowest non-zero setting. Try the coil: turn it on and leave it on. If it actuates and then returns to the rest position after about a second, the power is too low, so try the next higher setting. Repeat until the device actuates and remains actuated as long as the port is on.

Caution: always monitor the device for heating! Once you find a setting that's high enough to keep the device actuated, you have to verify that it's not *too* high a setting that overheats the device. It's possible that some devices don't have a safe operating zone at all - that is, a power level where they'll stay actuated and won't overheat. You should be able to monitor the device by touch: if it gets hot to the touch, turn it off immediately, since it's heating too rapidly and will probably overheat if you leave it on. If you can leave it on for a couple of minutes without having it get hot to the touch, it's probably in thermal equilibrium, meaning it should be safe to leave on in that state indefinitely.

Flipper button feedback control

One question that a lot of new pin cab builders ask on the forums is: how do I make the flipper buttons control the feedback solenoids for the flippers? In other words, how do I wire it so that pressing the flipper buttons activates the flipper solenoids?

There are two ways to go about this. The way that most people do it these days, which I consider the correct way, is to let DOF handle it. Happily, this is also the easiest approach, because it doesn't require any extra wiring or any extra configuration beyond the normal DOF setup you're going to do anyway.

The older way of handling this, which people did before DOF came along, was to hard-wire the flipper buttons to the flipper solenoids. This might seem simpler at first glance than the DOF approach, since it's just a directly wired connection. But it's actually more complicated to build! Some extra parts are required because you have to isolate the higher voltage that the solenoid uses from the low voltage that the key encoder uses. So DOF takes less work and fewer parts, assuming you're going to use DOF anyway. The only reason to use the hard-wired approach is if you're not going to include a DOF controller.

The thing I particularly like about the DOF approach, apart from its simplicity, is that it does a better job of simulating real pinball machines. If you think about how a real pinball machine works, the flippers only fire when the game allows them to fire - not between games, not after a TILT condition, etc. If you use the hard-wired approach, the flipper solenoids will end up firing every time you press the buttons, so the feedback effects won't always match the on-screen simulation. With DOF, they'll always match.

The modern way: let DOF handle it

If you're using Chapter 46, DOF, you don't have to do any extra work for flipper effects. Just set up DOF, and the flipper solenoids will work just like all of the other DOF effects. DOF will take care of activating the flipper devices whenever the on-screen flippers flip in a DOF-enabled game. The wiring for the flipper buttons and solenoids is exactly the same as for every other type of button and every other type of feedback device: wire the flipper buttons to your key encoder, just like all of your other buttons, and wire the flipper solenoids to your output controller, like your other feedback devices.

If you're trying to picture how the flipper solenoids actually operate with a DOF setup, here's the sequence of events:

- You press the flipper button, closing its switch
- The flipper button switch signals the key encoder
- The key encoder sends a "Flipper Button Pressed" keyboard event to the computer
- Visual Pinball (or whatever simulator you're using) gets the "Flipper Button Pressed" key event and fires the simulated flipper in the game
- DOF sees the simulated flipper event, and sends a FLIPPER SOLENOID ON command to the output controller
- The output controller activates the output port wired to your flipper solenoid
- The flipper solenoid fires

This approach is better than hard-wiring the solenoid for two reasons. First, it keeps the wiring simpler. The flipper button switch is only wired to the key encoder, and the flipper solenoid is only wired to the output controller - exactly the same wiring as for all of your other buttons and feedback devices. Second, it allows full software control over the flipper. That's the way it works on a real machine: try walking up to a real pinball machine some time and pressing the flipper button while GAME OVER is showing on the display. Does anything happen? No, of course not: the flippers are dead when a game isn't in progress. By wiring the flipper solenoids through the output controller, you'll get exactly the same effect. Hard-wired flipper solenoids would fire every time you press the buttons, whether or not a game was in progress.

What about latency? When DOF was new, and people started switching over to the DOF approach, there was some worry about the extra software steps adding a noticeable lag time between pressing the flipper buttons and hearing the feedback effects. Fortunately, it turns out that DOF is fast enough that this isn't a problem. You could measure the added latency using lab equipment, but it's not enough to be perceptible to a human. The human nervous system has a latency perception limit of about 30 milliseconds, and DOF in a properly working system is comfortably faster than that.

The old-fashioned way: hard-wire the buttons to the solenoids

If you're using DOF, you should ignore this section, because there's no reason to even consider hard-wiring the flipper buttons to the flipper solenoids. However, if you're not using DOF, you can get some very basic feedback effects by wiring your flipper buttons to your flipper contactors or solenoids, so that pressing the buttons directly activates the solenoids.

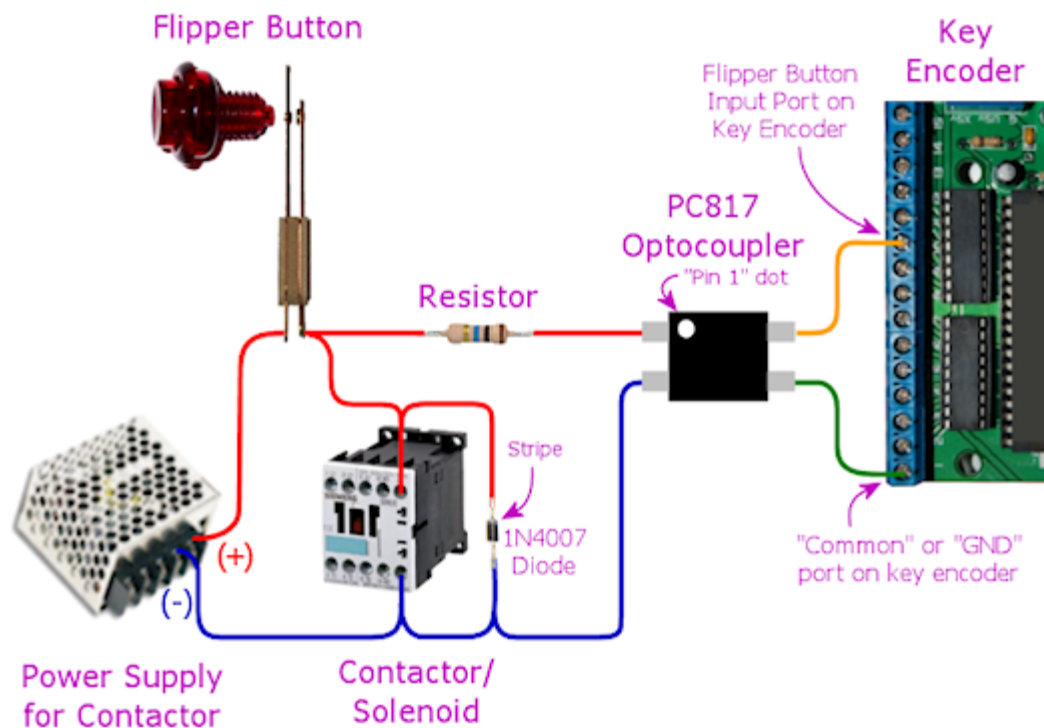
At first glance, it seems like it would be dead simple to wire the buttons to the solenoids. The complication is that you also need to wire the buttons to your key encoder, so that pressing the buttons sends key presses to the PC, to operate the simulated flippers in Visual Pinball and other pinball software. That dual wiring makes things more complex because of the different voltages used in the key encoder and the contactor or solenoid. The higher voltage needed for the contactors or solenoids would damage or destroy the key encoder if you wired both of them together directly. Instead, we have to isolate the two voltages, so that the solenoid voltage doesn't damage the key encoder.

To isolate the voltages, we can use something called an "optocoupler". That's a sort of solid-state relay that lets the voltage in one circuit control a separate circuit with a different voltage, while keeping the two voltages separated and isolated from one another. Which is exactly what we need in this case.

If you've tinkered with electronics before, you might be more familiar with regular mechanical relays, and you might be tempted to go with a mechanical relay to keep

things simpler. That would work, but I don't recommend it, because mechanical relays are fairly slow. Slow enough to cause an annoying amount of lag time in the software every time you press the buttons. Mechanical relays also tend to wear out with heavy use. Optocouplers are better because they're very fast and last a lot longer.

Here's the basic circuit design that should work for any of the common key encoders.



If you trace through the circuit, you'll see that the flipper button is wired so that it activates both the contactor and the optocoupler when pressed. Activating the contactor makes it fire, so you'll get the flipper feedback effect whenever you press the button. Activating the optocoupler completes the circuit on the key encoder side, so pressing the button will also send the key press for the flipper button. So whenever you press the button, you'll simultaneously hear the flipper feedback effect and send the key press to the PC. The optocoupler is the key to making both devices work together, because it uses the higher voltage on the solenoid side to control the key encoder side, but it does so without allowing the higher voltage to reach the key encoder side.

The diode is needed to protect the rest of the circuit from the voltage surge effect from the contactor or solenoid. A diode like this is needed for every device with a coil or motor. See Chapter 53, Coil Diodes for more details.

Parts selection: The PC817 optocoupler shown is only an example. Just about any standard optocoupler should work, if there's another type you prefer or that you can find cheaper.

The resistor has to be selected according to the contactor's power supply voltage and the type of optocoupler you're using. Here are the appropriate resistor sizes for the PC817 and common power supply voltages for the contactor:

Voltage	Resistor
5V	180 ohms, 1/8 Watt or higher

12V	560 ohms, 1/4 watt or higher
24V	1200 ohms (1.2K), 1/2 watt or higher
48V	2400 ohms (2.4K), 1 watt or higher

If you're using a different optocoupler chip, the required resistor values might vary slightly from the table above, so you should calculate it using an LED resistor calculator. See Chapter 52, LED Resistors for help with this - that chapter includes an interactive calculator that will figure the right resistance value for you. You'll have to look up the following numbers in the data sheet for your optocoupler, which you can then plug into the calculator in the LED Resistors chapter:

- I_F , the Forward Current for the optocoupler LED
- V_F , the Forward Voltage (also known as Voltage Drop) for the optocoupler LED

Plug those two values into the calculator in the LED Resistors chapter, along with the power supply voltage for your contactor, and the calculator will tell you the size of resistor to use.

60. Replay Knockers

The replay knocker is the device in a real machine that makes the loud knock noise when you win a free game, exceed the replay score, etc. This is implemented in the real machines with a solenoid coil positioned in the backbox so that its plunger strikes a metal plate when fired.

The knocker isn't used all that often, and all it does is make a noise, but I still consider this a must-have toy. When the knocker does fire, it's always at particular dramatic points during play, so it really demands the same dramatic sensory effect as the real thing. Recorded audio played back through speakers is a pale imitation.

Parts

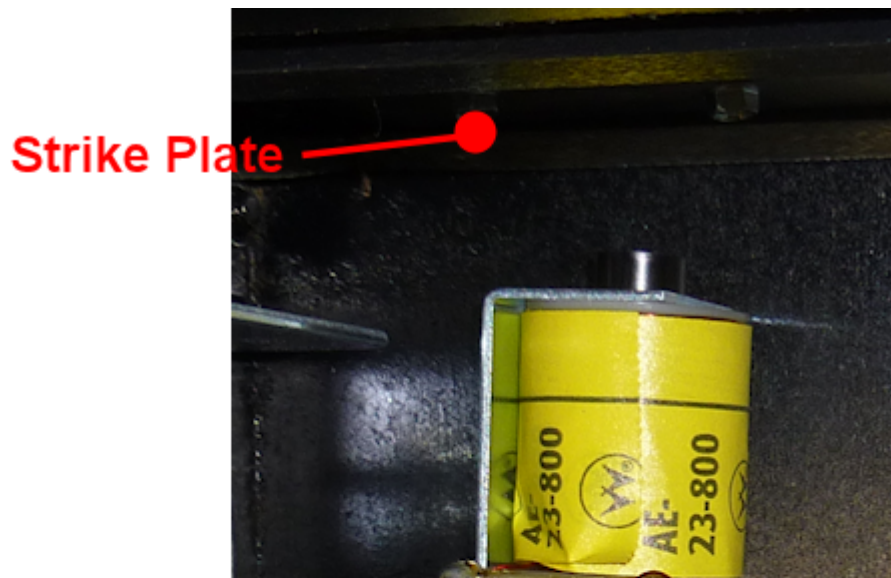
Knockers are pretty simple mechanisms, so it wouldn't be hard to improvise a DIY version using a push-type solenoid. But there's not much point to that, since you can just buy a real one for about the same money it would cost to build a DIY version. And the real ones come as full assemblies, so they're easy to buy and install.

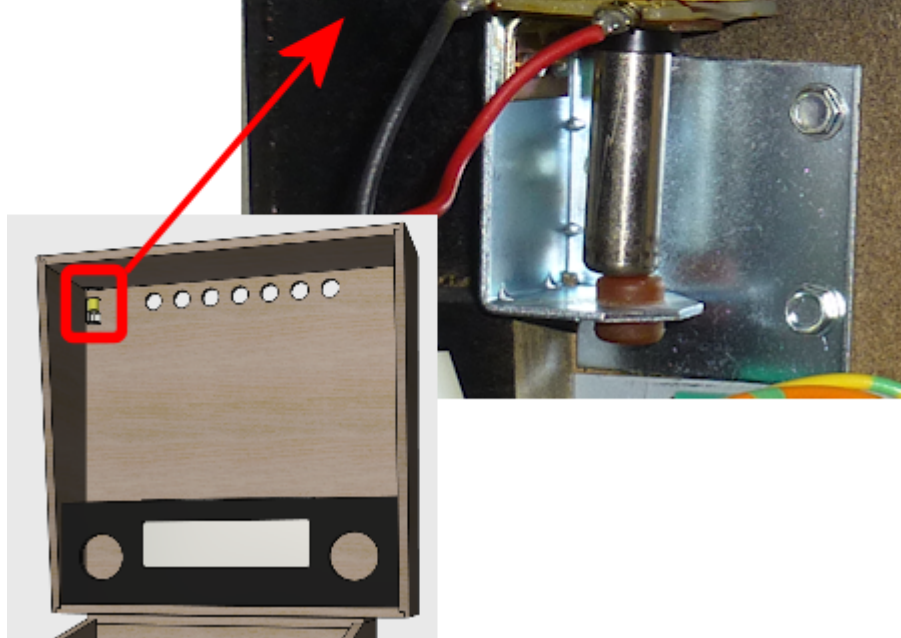
There are two parts to buy, both available from pinball parts vendors like Pinball Life and Marco Specialities:

- The knocker assembly itself. Look for Williams/Bally reference B-10686-1. You might be offered a choice of coils. Any of the options available from your vendor should work, but the recommended one is AE-26-1200. These sell for about \$25.
- A knocker strike plate. This is a small steel plate that the knocker plunger hits, to enhance the sound effect (and protect the backbox area where the plunger hits). You can substitute a similar metal plate from Home Depot, such as a steel backing plate or tie plate, but the pinball part is cheap (about \$3), so you might as well add one to your cart as long as you're ordering the knocker assembly.

Installation

Modern replay knockers are designed to be mounted vertically in the backbox, on the back wall near the top, so that the plunger strikes the ceiling of the backbox. They have to be installed vertically because the assemblies don't have springs; they depend on gravity to pull the plunger back to the rest position after firing.





Install the knocker as shown above, so that the top of the knocker assembly is about 1" below the ceiling of the backbox. Mount the strike plate on the ceiling just above the knocker assembly.

The knocker assembly should come with the plunger already inserted through the coil, but it can fall out. If it has fallen out, insert it so that the white plastic tip is facing **up**. The white plastic tip is the part that hits the strike plate. Many people find this surprising - it seems like the metal end should be the hammer end. That's understandable but wrong. The metal end is actually the magnetic element that gets pulled into the coil, so it should be facing down.

Power supply

Modern replay knockers are designed to run on 50V DC power. They'll run on lower voltages, down to about 24V, but the power of the effect will be correspondingly reduced.

50V supplies are expensive difficult to find, but you can find relatively inexpensive 48V supplies on eBay. If your budget can tolerate it, I'd buy a dedicated 48V, 5A supply, which are available for about \$20. If you're using any other real pinball assemblies, such as flippers, slingshots, or bumpers, you'll need a 48V supply for those anyway, so you can just attach the knocker through that.

If your budget is tight, you can use an existing 24V supply, but the effect will be a bit weak. You might consider buying a "DC to DC step-up converter" on eBay to boost the power as high as possible; converters that will take the power to about 40V are available in the \$10 range. But given that a dedicated 48V supply is only a little more expensive, I think that's a better option.

Caution: 48V is a hazardous high voltage. Be careful to turn off power when working inside the cabinet once you have a 48V supply installed. Note that there will be a live +48V voltage to the knocker's red wire any time the power supply is connected, *even when the knocker isn't firing*, since the knocker is controlled through the black wire.

Wiring

A new knocker assembly usually comes with a diode already installed, and two wires,

one red and one black. Connect the red wire directly to the positive (+) power supply voltage. Connect the black wire to an available output port on your output controller.

If your coil *doesn't* come with a diode already installed, you must add your own, to protect your output controller and other electronics from interference from the magnetic field from the coil. See Chapter 53, Coil Diodes.

The Williams/Bally knocker assemblies come with a Molex 3-pin .093" cable connector. You can cut this off and solder your own wires if you like, or you can buy a matching connector. The matching connector is available from pinball vendors and from electronics companies like Mouser:

- Molex .093" 3-position receptacle
- Crimp socket (female)

If you're using the Pinscape expansion boards, connect this to the "Knocker" pin on the JP9 pin header. This is a "timer protected" output, to prevent the knocker coil from getting stuck on. That protects the coil from overheating in case of software failure. See Chapter 54, Coil Timers.

If you're using an LedWiz, you'll have to use a booster circuit of some kind. The replay knocker will draw 3-4A when activated, which exceeds the 500mA (0.5A) maximum for an LedWiz port. You can connect the knocker directly to the Pinscape expansion board's dedicated "Knocker" output, or directly to any Power Board or Chime Board output port.

DOF Setup

In the DOF Config Tool, go to the Port Assignments page. Find the port number where you wired the replay knocker coil. Assign this to "Knocker".

61. Shaker motors

The shaker motor is one of my favorite feedback toys. It does what the name suggests: it shakes the cabinet when activated. When done properly, the effect is a deep rumble that reverberates through the floor like an earthquake. Appropriately, the first arcade pinball that featured a shaker motor (as far as I know) was the earthquake-themed *Earthshaker* (Williams, 1989).

Stern has enthusiastically embraced shakers in recent years, offering them as optional equipment in nearly all of their titles since 2010. Shakers were rare in the arcade before that, appearing only in a few titles where there was some kind of thematic connection, such as *Earthshaker* (for obvious reasons) and *Red and Ted's Road Show* (for the demolition theme). I think Stern realized that the tactile effect is a lot of fun for its own sake, even when it doesn't particularly tie in to the theme, which is exactly why shakers are popular in virtual cabs. You can profitably work it into pretty much any game, even older games that didn't have shakers in their original arcade versions. The DOF Config Tool database takes this to heart, liberally adding shaker effects to lots of older tables - as of this writing, the DOF database uses the shaker in 237 (!) games. Of course, DOF also faithfully reproduces the authentic usage in games like *Earthshaker* that actually did have shakers originally. If you include a shaker, it will get plenty of use.

Theory of operation

A shaker is simply a motor with an unbalanced weight attached to the shaft. It's like a washing machine a big heavy load on spin cycle.

Off-the-shelf pinball shakers

You can buy ready-to-use pinball shaker units from the pinball parts vendors, including Marco Specialties and Pinball Life. Off-the-shelf kits sell for \$100 to \$200, depending on the target machine type.

The different kits currently available all use exactly the same shaker mechanism (as far as I can tell). The only reason there are so many different kits is that each comes with an interface board that works with a particular type of Stern machine. For a virtual cab, you actually don't need an interface board at all, because your DOF output controller will perform that function instead. So any of the Stern kits should work, and I'd just buy the cheapest one you can find, currently around \$100. When the kit arrives, just throw away the Stern interface board, and plug the motor directly into your feedback device controller; see Wiring below.

Off-the-shelf vibration motors

There's another off-the-shelf option that's less expensive than the Stern shaker kits: replacement "vibration motors" for massage chairs and similar appliances. Some people on the forums have tried these and reported generally positive results.

You can find these on Amazon, eBay, and Aliexpress. Some are as little as \$20. They're all no-brand parts from third-party sellers that come and go overnight, so I can't give you any specific model numbers or product links, but you can find options by searching the retail sites for **vibration motor**. Look for devices that have two counterweights and a speed of 4500 RPM or less. Lower speeds are better - higher speeds will feel too much like buzzing.

Be aware that there are smaller and larger versions of these motors. The smaller ones are more like cell phone buzzers, so skip those. You want one of the larger ones, sold for applications like massage chairs.

Even the largest vibration motors I've found still use smaller counterweights than the Stern units, so they'll produce a weaker shaking effect. I've seen a couple of negative comments about this from virtual cab builders. But stronger isn't necessarily better; there's an element of taste to this, plus, a less powerful unit might be more suitable if your cab is smaller and lighter than a full-sized pinball machine.

DIY designs

Shakers are simple enough that you can build one yourself for about \$50 to \$100 in parts. That's not much of a cost savings compared to buying one of the ready-made Stern kits, and it's actually more expensive than buying an OEM vibration motor, but some people might want to build their own just for the enjoyment of the project. A DIY shaker also gives you more control to tweak the effect to your liking, by adjusting the amount of weight and the geometry.

Here are some detailed DIY plans you can find online:

- Appendix 15, My shaker design
- Darkfall's plans on vpforums
- RacerRik's plans on Pinside
- Waldo34's plans on Pinside (a variation on RacerRik's plans)

The two Pinside plans closely replicate the mechanical design of the Williams and Stern shaker units (which are themselves nearly identical), so they yield nicely professional-looking results, but they're challenging to build because they require cutting, drilling, and thread-tapping steel bar stock. They also require some sheet-metal work. You need some specialized tools for all of that metal working, and the steel bar stock itself is rather expensive. Darkfall's design is easier to build because it doesn't require any metal fabrication, and it doesn't require any especially exotic parts. The first shaker I built was based roughly on Darkfall's design, and it works nicely. My only criticism is that its method for attaching the weights to the motor shaft is a bit jury-rigged and seems precarious - although, to be fair, the one I built has held together for many years now. My own design is an attempt at a more elegant and robust way to attach the weights, while still being easy and inexpensive to build. Almost all of the parts required in my plan are common hardware store items, and the only things you have to fabricate yourself are some simple plywood pieces.

Designing your own shaker

If you prefer to develop your own DIY plan, here are some parameters you can use as a starting point, based on what seems to work in practice for the commercial units and the DIY designs:

- Around 3000 RPM for the spin rate
- About 350 gram-centimeters (5 ounce-inches) of off-balance weight

The "gram-centimeters" figure refers to the result of multiplying the total weight of the counterweights by the distance from the shaft to the center of mass of the counterweights. It's the product of these two numbers that matters, so you should get the same effect from a 100 gram weight at 3.5cm or a 350 gram weight at 1cm,

for example.

For reference, the counterweights in most of the commercial shakers are steel bar stock, 3/4" wide x 3/4" thick x 2" long, weighing 140g each. The shaft is placed 1/2" from the center of the long dimension (2").

Selecting a motor for a DIY unit

The motor is the hardest part to find when building a DIY shaker unit. There are hundreds (maybe thousands) of types of small electric motors available online, but most are unsuitable for one reason or another - they're too big, too small, use too much power, etc. It's especially hard to find all of the right properties in a dual-shaft configuration, since there just aren't that many dual-shaft motors available. A dual-shaft motor isn't an absolute requirement, but all of the DIY designs call for one, because it's easier on the motor's bearings to divide the weight into two smaller chunks than to have a single large counterweight.

The easiest way to source a suitable motor, by far, is to buy one that's marketed specifically as a replacement motor for a pinball shaker unit. Those are more expensive than comparable generic motors from eBay or Aliexpress, but you can be confident that they're suitable for the task, and you won't have to comb through hundreds of listings looking for the right specs. The pinball parts vendors currently (2022) stock several such motors:

- Pinball Life - replacement motor for Stern shaker kits
- Marco Specialties - replacement motor for Stern shaker kits
- Marco Specialties - replacement motor for 1990s Williams shakers, part 14-7951

If you want something less expensive, or you just enjoy the hunt, here are the properties to look for in a generic motor:

Parameter	Range	Comments
Configuration	Dual-shaft (also called double-shaft)	This means the rotating shaft sticks out at both ends of the motor, allowing you to attach a weight at each end.
Voltage	12VDC	Most DC motors will run on a wide range of voltages (e.g., 6V-24V), so just make sure that the power supply you plan to use is within the voltage range listed for the motor. 12VDC is usually the most convenient power source in a pin cab.

Power	20W-50W	The power rating gives you an idea of the mechanical force that the motor can muster, and the load it places on the power supply. A motor rated for 20-50W should have enough mechanical power for strong shaking, with a power draw at 12V of 4A or less. That's in the range that most ATX power supplies and high-power DOF controllers can safely handle. Motors with higher wattage ratings might overload your power or control systems.
Speed (no load)	3000-4500 RPM	Higher speeds might feel more like buzzing than rumbling.
Shaft diameter	4mm-6mm or 1/4"	This won't affect the feel of the effect, but smaller shafts might bend under the force of the shaking.

For reference, here are some specific motors that virtual cab builders have used successfully in the past. It's probably impossible to find anything on this list that's more than a couple of years old - the manufacturers generally seem to do a single large production run for each model, and once that's sold out, they're gone. But it might still be worth checking, since you can sometimes find the older models as remnants and used items on eBay.

- Generic 775 12V/24V 50W dual-shaft motor (available on Amazon and Aliexpress as of June 2022; note that the "775" is **not** a model number or brand, but is simply a standard designation for the motor's dimensions; many other "775" motors are available that look similar but which have very different electromechanical characteristics)
- Pittman 9414H255-R2 (last seen around 2015)
- Buehler 480-0211-01 (last seen around 2015)

If you know of any other specific motor models that I should add to the list, please let me know.

Historical note: The motor in the original Williams shakers was labeled Johnson HC970. Unfortunately, that's not very helpful for finding replacement parts, because Johnson Electric apparently used this designation for a number of different OEM parts that they custom-manufactured for different customers. I've seen a couple of reports on the forums from people who found Johnson HC970 motors on eBay that turned out to be unsuitable.

Converting a regular motor to dual-shaft

Most shaker unit designs call for dual-shaft motors. But dual-shaft motors are relatively rare. This is the main thing that makes it so difficult to source a motor for a DIY shaker unit. One way around this is to buy a regular motor, and convert it to a dual-shaft configuration.

This isn't exactly an easy solution. It requires taking the motor apart, modifying its inner workings, and putting it back together in working order. You'd have to be comfortable doing major repair work on mechanical appliances to contemplate this,

and you should only consider it if you can tolerate the risk that you'll destroy the motor in the process. Most small DC motors are not at all designed to be taken apart by the end user. Attempt this only with a cheap motor that you don't mind replacing if you end up breaking it.

The basic idea is to disassemble the motor and replace its original shaft with a longer one - long enough to stick out from both ends of the motor body. The full procedure is beyond what I can document here, especially because motors aren't all exactly the same on the inside. There are some Youtube videos on the subject that you might find helpful - try searching for **convert motor to dual shaft**. Here's just a rough outline of the process:

- Open the motor casing. This usually requires using vise grips or pliers to forcibly bend back some metal tabs that hold the two halves of the main outer casing together. (I did warn you that most motors aren't designed to be taken apart!)
- Remove the whole armature/shaft assembly. Before you do, you might want to take a bunch of close-up pictures, to record how the brushes, springs, bearings, etc. are all arranged, so that you can get it all back together later. Beware that the brushes are usually spring-loaded, so small parts might go flying when you remove the shaft. You might want to work the motor inside a box or plastic bag so that flying parts don't get lost.
- Remove the shaft from the armature. In most motors, the shaft is simply press-fitted into the armature (the part with the wire coil windings), meaning that the only thing holding it in place is friction. Removing it just requires some brute force to pull or hammer it out. The fit is usually very tight, so this can take quite a bit of force.

If the shaft is long enough to start with, you might not need to replace it at all. You might be able to simply reposition the armature along the shaft so that some of the shaft sticks out the back of the motor.

- Assuming you do need to replace the existing shaft with a longer one, you'll now need to cut a piece of round steel rod to the desired length to make the new shaft. You'll need steel rod of exactly the same diameter as the original shaft. You can buy steel rod stock from Amazon or a hardware store. Cut it to the desired length, probably about 1" to 2" longer than the original shaft, using a cut-off wheel, hacksaw, etc.
- Apply some thread-locker to the new shaft so that it'll stay in place. Insert the new shaft into the armature, and position it so that enough material will stick out on each side of the motor body to provide attachment areas for the counterweights on both ends. As with removing the shaft, this is a brute-force matter of pushing or hammering it into place.
- Put the motor back together and bend the casing lock tabs back into place.

Enclosure

A shaker needs a sturdy cover that will contain the weights in case they ever come loose. A detached weight could become a high-speed projectile that could cause a lot of damage if not contained.

The off-the-shelf pinball shaker units all come complete with their own plastic enclosures, so that's another point in their favor. If you build a DIY plan, or buy one of the generic vibration motors mentioned earlier, you'll need to provide your own enclosure. I'd suggest a simple hand-built plywood box. (A clear acrylic cover is a nice enhancement if you build your own box, since it lets you safely observe the

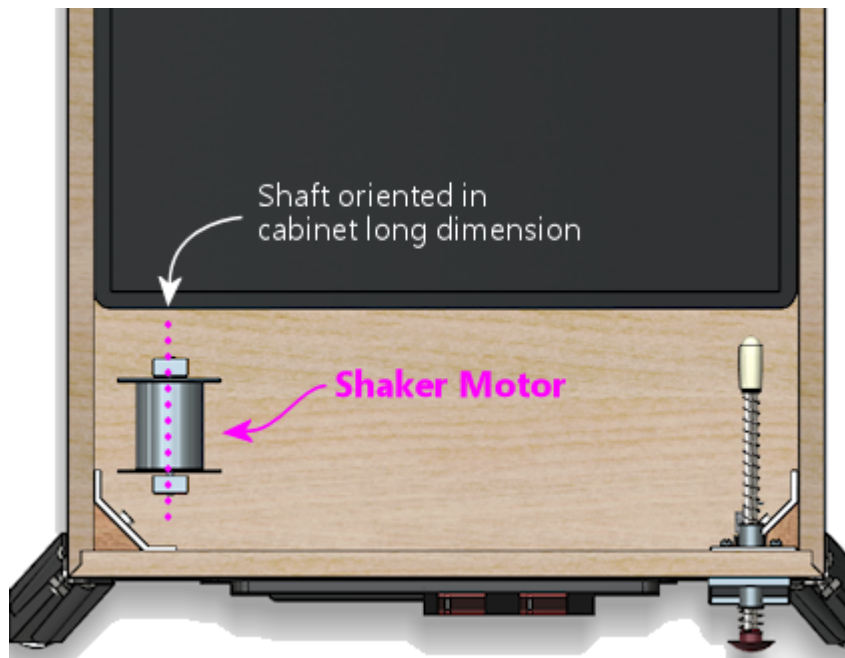
shaker in operation whenever you want to check that it's working properly.) You could also re-purpose a plastic food container or storage box, if it's strong enough to contain an ejected weight.

Where to mount

On the real machines that have shakers, the units are installed somewhere near the front of the cabinet, but the exact location varies by title. I think it's just a matter of what's convenient in that machine's cabinet layout. Most of the recent Stern machines place the shaker in one of the front corners, alongside the coin box. In other machines, the shaker is behind the coin box, either centered left-to-right or off to one side.

I don't think there's much difference in effect from the different placement options, so put it wherever you have room. The only imperative, I think, is that it's better to keep the shaker close to the front of the cabinet, since that's where the player is.

In every commercial pinball setup I've seen, the motor shaft is oriented parallel to the long axis of the cabinet. I think this orientation does matter, since it creates a side-to-side shaking motion. That's probably the best direction for maximizing motion transfer, given the cabinet's proportions.



Interaction with nudge devices

Some people have trouble with the nudge device picking up accelerations from the shaker. The nudge device is an accelerometer whose whole purpose is to detect cabinet motion, so it necessarily detects the motion that the shaker causes the same way that it detects motion from the player's nudges.

The shaker's motion shouldn't interfere with game action, even though it registers on the accelerometer. The motion from the shaker is different from nudges in that it's symmetrical, it's faster, and it's lower amplitude. The back-and-forth shaking motion should essentially cancel out on the time scale of manual nudging. Player nudges, in contrast, tend to be bigger and in one direction, and they happen on a longer time scale.

If you're having a problem with your nudge device going crazy when your shaker is

on, there are two ways to address it:

- Turn down the intensity of the shaker
- Turn down the sensitivity of the nudge device

Before taking either action, you should ask yourself: What would a real pinball machine do? In other words, would that same exact shaker motor affect the ball on a real pinball machine the way it's affecting your virtual game? If the answer is yes, then the shaking effect is probably too intense, and the solution is to reduce the shaker motor speed. Shakers on the real machines don't send the ball flying around. If your shaker is at a level where it wouldn't affect the ball on a real machine, but it's affecting the virtual game to a troublesome degree, then the problem is in your nudge sensor calibration: you've turned up the sensitivity to unrealistic levels.

I'd always start by adjusting the shaker to produce a tactile effect you like, ignoring its effect on your nudge sensor. The easiest way to adjust the shaking intensity is to adjust the speed of the motor, either by adjusting the voltage or by adjusting the DOF PWM parameters; see "Speed adjustment" below. If you built a DIY shaker, you might also be able to change the shaking intensity by increasing or decreasing the amount of weight in the counterweights.

If the shaker is still causing excessive nudge interference after you've adjusted the intensity of the tactile effect to your liking, the next step is to adjust the sensitivity of your nudge device. You might balk at this suggestion, but go back to that question about real machines: does the ball fly around wildly on a real machine when the shaker is on? No, it doesn't. The thing is, most virtual cab builders initially set their nudge devices to be far too sensitive. This is natural - you want to see an immediate and obvious effect from the new toy. But it's easy to overdo this. I always urge new cab builders to go find a physical pinball machine and play around with it for a while, to see how the ball reacts in reality. If you mostly play virtual pinball, you'll probably be surprised by how "dead" a real machine feels when you nudge it. On a real machine, a ball trapped on the flipper will *not* fly up off the flipper when you give the machine a little push, the way many people think it's supposed to work in Visual Pinball. If you want the shaker to coexist peacefully with your nudge device, you'll have to adjust your nudge device sensitivity so that it's closer to the subdued response a real ball would show.

One more thing: please don't use "dead zones" in your accelerometer setup. Dead zones are terrible. They make erratic behavior even more erratic because they create a non-linear "cliff" where there's no response at all at one level, and suddenly a huge response just a hair above that. It's better to stick with purely linear settings, like the "Gain" settings in Visual Pinball.

For more on nudge adjustments, see Chapter 36, Nudge & Tilt.

Speed adjustment

The amount of shaking you get out of your motor is a function of the weights, the distance they are off-axis, and the speed of the motor. It also depends on factors that aren't related to the motor itself, like where it's mounted in the cabinet, the overall weight of your cabinet, the stiffness of the legs, and the construction of the floor the machine is sitting on. The same shaker motor will produce somewhat different effects in different cabinets.

Of all of these factors, there's one that we can easily change: the speed of the motor. Once you have everything set up, you can fine-tune the effect by adjusting the motor speed up or down.

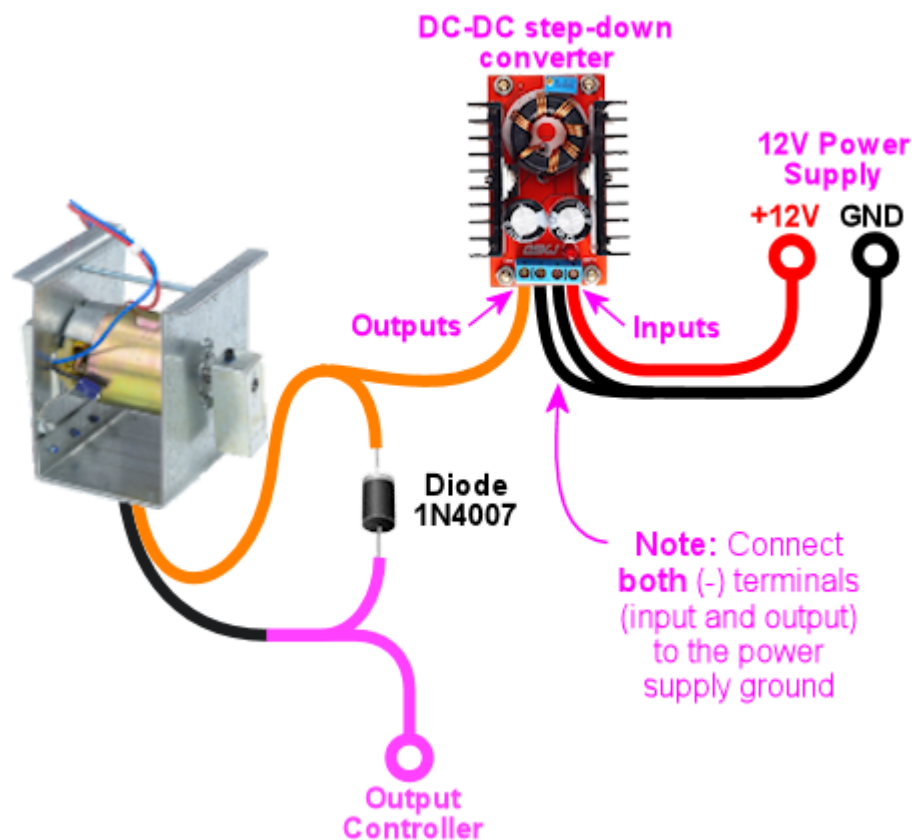
There are two ways to control the motor speed: adjusting the voltage of the power supply, and adjusting the power in software via PWM control (Pulse Width Modulation - a method of varying the power level in a digital circuit by rapidly switching the power on and off).

Adjusting the voltage: Most DC motors will work with a range of power supply voltages, spinning faster at higher voltages and slower at lower voltages. You can take advantage of this to adjust the motor's speed by varying the power supply voltage.

The best way to control the voltage is to connect an adjustable voltage regulator between the motor's power input and the 12V supply, as shown in the diagram below. Look for a "DC-to-DC step down converter" on eBay, and find one that has a dial or set-screw that sets the output voltage. This will let you adjust the voltage to anywhere between 0V and 12V, so that you can fine-tune the motor speed to your liking.

Select a regulator that has an amperage rating high enough for the motor (equal to or higher than the amperage the motor draws).

Note that a motor always has a minimum voltage level, below which it won't have enough torque to start spinning. The exact minimum is something you have to determine experimentally, since it depends upon the amount of load the motor is driving. If you try a low voltage and the motor stalls, don't leave the power on very long, as a stalled motor can quickly overheat. When experimenting to find the right voltage, it's safest to start with the dial turned up to the high end (12V), and gradually reduce the voltage until the speed is to your liking.



Typical wiring for a shaker motor with a DC-to-DC step-down voltage regulator to adjust the voltage supplied to the motor. Check your converter for the correct input/output terminals (don't rely on the illustration - your converter might have a different terminal arrangement). Note that the converter's (-) input and output terminals are both connected to the power supply ground.

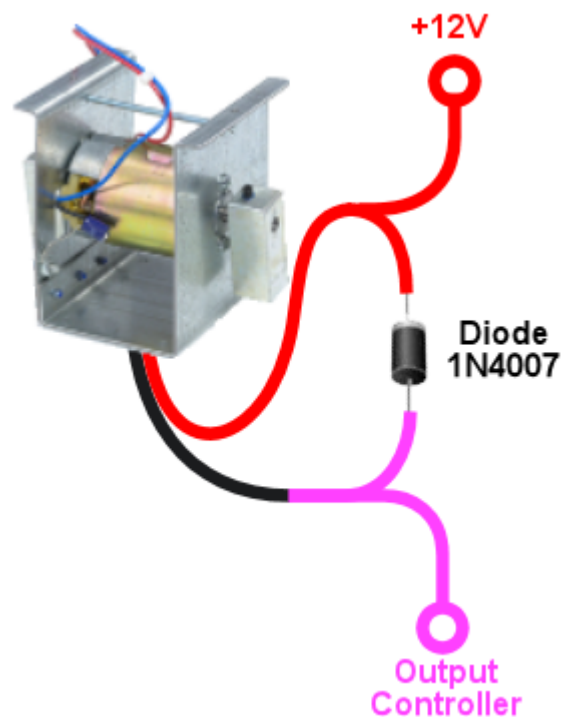
Adjusting with PWM: The DOF Config Tool lets you set the power range for the shaker motor. Go to the Port Assignments page, and look for the "Shaker Motor" section on the right side of the page. This will let you set a maximum intensity, on a 1-48 scale. The default is level 48, to run the motor at full speed. Lower settings should slow down the motor; you can try different settings until you find a speed you like.

Most motors have a minimum power level they need to operate at all, so there will probably be some minimum PWM value below which your motor won't start. Don't leave the power on to the motor if it's stalled, as that can overheat it. When experimenting to find the right PWM level, it's safest to start at the maximum (48) and work down from there until the speed is to your liking.

Note that the PWM control only works if you're using a PWM-capable output controller, such as an LedWiz or Pinscape Power Board. PWM won't work with a relay-based controller such as a Sainsmart. You'll have to use the voltage adjustment approach with those.

Wiring

Follow the general wiring plan for any output device (Chapter 47, Feedback Device Wiring). Connect one terminal of the shaker motor to the positive (+) power supply voltage (usually 12V). Connect the other terminal to an available port on your output controller.



A diode is required, to protect your output controller and other electronics from interference from the motor's magnetic field. See Chapter 53, Coil Diodes. If you're using a pre-built shaker assembly, it might or might not already have a diode installed; if you don't see one, assume there isn't one and add your own.

If you're using one of the off-the-shelf Stern shaker kits, and it came with an interface board for a real pinball machine (for example, a Stern SAM connector or a SPIKE connector), you won't need that interface board to use the motor in your virtual cab. Those boards are designed to interface to the specific electronics found in the various Stern machines, so they're not relevant to a virtual cab. We just need the

motor itself.

If you're using the Pinscape expansion boards, you can connect the shaker directly to any MOSFET Power Board port. If you're using an LedWiz, don't connect the motor directly, as it will draw too much power for an LedWiz port; you'll need some kind of booster or amplifier circuit. See "Power limits and boosters" in Chapter 50, LedWiz Setup.

H-bridges: If you read through old posts on the forums, you might see people say that you need an "H-bridge" to control a shaker motors. You don't. The idea got embedded at some point in the group consciousness because someone read about it on an Arduino forum and it got repeated a lot. H-bridges are useful if you need to control a motor bidirectionally, which is something that Arduino robotics hobbyists often want, but isn't necessary with a shaker motor. A shaker motor just needs to run in one direction, so you can connect a shaker directly to any feedback device controller port that can handle the motor's power level. If you're using the Pinscape expansion boards, you can run a shaker motor directly from any Power Board port. If you're using an LedWiz, you'll need a power booster circuit, the same as you'd use for any other type of high-power device like a solenoid or coil.

LedWiz hacks: You might also see old posts with some really scary LedWiz modifications involving soldering wires to IC pins on the LedWiz board. Ignore those. They're based on that old H-bridge notion. If you're using an LedWiz, you will need some kind of booster circuit, but that's something you can add on externally. There's no need to modify the LedWiz in any way. You just need the same sort of booster circuit you'd need for any other high-power device, as described in Chapter 50, Power limits and boosters in Chapter 50, LedWiz Setup.

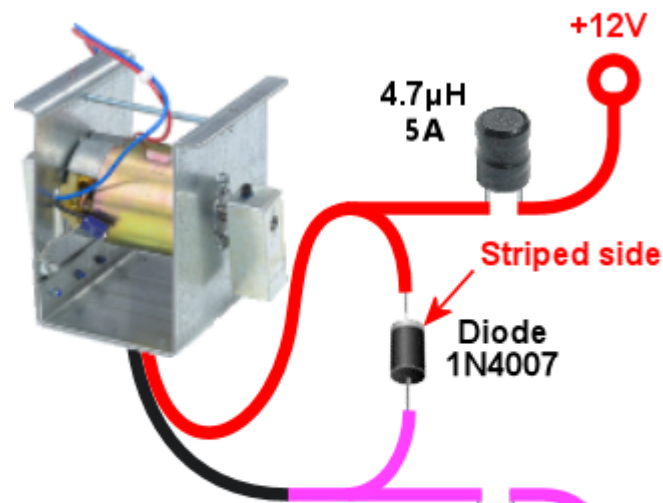
Electrical interference

Motors produce a lot of electrical noise, which can feed back into the power supply wiring and propagate into your system's logic circuits, such as the feedback controller and even the PC motherboard.

Diodes are a must, as already mentioned, because they help suppress the high-voltage transients that motors produce.

For some motors, diodes might not be enough. If you start seeing noticeable glitches when the motor is running, such as USB disconnects or random keyboard input, you might need some additional filter components.

The typical noise filtering for a motor is to add a pair of "chokes" (a type of inductor), in series with the wiring to the motor. Add one on the power input to the motor and one on the connection to the feedback controller.





Try a 4.7 μ H inductor, with an amperage rating equal to or higher than your motor's operating current. Here's an example part from Mouser that should work well:

Coilcraft DR0608-472L 4.7 μ H, 5.8A radial inductor - at Mouser

Inductors aren't polarized, meaning they don't have a special orientation when you install them. It doesn't matter which lead connects to the "+" side and which connects to the "-" side. (The diode, in contrast, has to be installed with its striped side going to the "+" voltage, as shown in the diagram.)

The inductors shouldn't be necessary on most virtual cabs. I haven't had a need for this with any motors on my machine. But it's something to try if you experience interference problems that seem to coincide with shaker activation. You can also add inductor filtering to any other motor-based devices that cause interference, such as gear motors, fans, and beacons.

Note that you should always keep the diode in place even when adding inductor filters. The inductors provide a different type of filtering and aren't a substitute for the diodes.

DOF Setup

In the DOF Config Tool, go to the Port Assignments page. Find the port number where you wired the shaker motor. Assign it to "Shaker".

At the right side of the page, you'll also find a section labeled "Shaker Motor" that lets you set the intensity range. If you're using a PWM-capable controller (e.g., an LedWiz or a Pinscape power board), this lets you set the range of power that DOF uses when the shaker runs. The intensity values are on a 1-48 scale, where 48 is the highest power. The default settings use the full available range. If you find that the shaking effect is too powerful when DOF activates it during game play, you can reduce the maximum intensity setting to slow down the motor. Similarly, if the motor seems too weak some of the time, or doesn't have enough power to start spinning in some cases, you can raise the minimum setting.

Note that the intensity adjustment won't work unless you're using a PWM-capable output controller. If the motor is connected through any kind of relay, such as a Sainsmart board, the DOF PWM adjustment won't work and you'll have to adjust the speed some other way, such as with a varying voltage supply (see "Speed adjustment" above).

62. Gear motors

A gear motor can be used in a virtual pin cab to re-create the noise made by the playfield motors found in many real pinball games. A gear motor is just a small DC motor with some step-down gears to reduce the shaft speed.

If you look at real machines through the 1980s and 90s, you'll find a surprisingly high percentage had some kind of moving playfield feature that was animated by a small motor: Thing in *The Addams Family*, the cannon in *Terminator 2: Judgment Day*, the magic trunk in *Theatre of Magic*, the drawbridge in *Medieval Madness*, and many more.

The point of adding a gear motor to a virtual cab is simply to replicate the noise those little motors make in the real games. The motor doesn't actually do any mechanical work, and you don't have to attach anything to its shaft; it's simply the noise of the motor itself that we're interested in. As with other noise-making devices, it's always more realistic to have something real making the noise than to rely on recorded audio played back through speakers.

I'd rank gear motors at the low end of the impact scale, as all they do is make a little noise - and that's actually a little artificial, since the motors in the real machines were almost never audible over the game's other sound effects. In virtual cab use, most people intentionally use motors that are louder than the real thing because they want to be able to hear the effect. So I'd give higher priority almost all of the other toys if you're economizing. But I'm in the minority on that; a lot of cab builders really like their gear motors. And if you're going for a decked-out cab with all the toys, by all means include a gear motor, as it does add a little something, even if the emphasis is on *little*.

Parts: You just need the gear motor itself, and whatever fasteners you need to mount it somewhere inside the cabinet.

Gear motors are available in almost infinite varieties, but there are two types most commonly used in pin cabs.

The first is small, cheap DC gear motors made for robotics (Arduino) use. Search eBay for "small 6V gear motor" and you'll find numerous options. The type favored by pin cab builders has exposed gears like the one pictured at right. The exposed gears are desirable because they make for noisier operation, which is what we're after here.



The second common type in pin cabs is replacement motors for windshield wipers in cars. Like nearly all automotive products, these operate at 12V DC, so they can be conveniently powered from a computer ATX power supply. One model that many people use (I use it on my own cabinet) is a VW OEM replacement part; look for part number 251955119191955113A on eBay or an auto supply store site.

Which to choose?

- If you want something cheap, use a robotics motor. You can find dirt-cheap options on eBay for those (under \$5).
- If you want something loud, use a robotics motor.
- If you want something more subtle, use a windshield wiper motor. They're designed to be quiet - passengers in the car don't want to hear them, after all. They produce a subtle, low frequency "hum" or "whir", vs the loud buzzy noise you'll get from a robotics motor.

I don't want to make "subtle" sound like a value judgment. This is a matter of taste, so there's no right or wrong here. The downside of making the motor quieter is that you can't hear it if you make it *too* quiet, and there's no point in installing it if you can't hear it. And the windshield wiper motors can in fact be too quiet. You might not hear it at all lot of the time, and when you do, the effect will be very understated. I personally think the understated effect is a lot more realistic, as (like I said above) you rarely hear the motors operating in the real machines. But that defeats the purpose of including it at all for a lot of people, so if you think you'll be disappointed with too much subtlety, you might prefer something noisy enough that you're sure to hear it.

The original gear motors in the real pinball machines that we're emulating were typically somewhere in between the robotics motors and windshield wiper motors in size, but more at the wiper-motor end of the spectrum. Most of the ones I've encountered in person are pretty much inaudible - you tend to hear the sound effects that the game plays when the motor is operating, not the motor. So in that respect, I think it's more authentic to use one of the quieter wiper motors.

Positioning: It's worth giving a little thought to where the gear motor will be mounted, since human hearing is pretty good at locating sounds in space. You want the motor noise to sound like it's coming from the playfield element it's simulating.

My recommendation is to mount the motor under the playfield TV, centered side-to-side and about 2/3 of the way back from the front of the TV. In the real games we're simulating, the motorized feature is almost always some major centerpiece element, and those tend to be towards the rear of the playfield. The rear center of the TV is a pretty good "average" of the locations where the elements we're simulating would appear in the real thing, so it will make the sound seem to come from about the right place most of the time.

There's no standard way to mount these motors. Just improvise something using fasteners from a home supply store. U-bolts are often helpful here.

Wiring: Make sure you're using a suitable supply voltage for your motor. An automotive windshield wiper motor should run on 12V; the eBay robotics motors might specify anything from 3V to 12V, depending on what you buy. DC motors will usually run happily on voltages lower than their rated voltage, with a corresponding reduction in speed. You just want to make sure the voltage is high enough to make the motor turn at all, since a completely stalled motor might overheat or overload your power supply. So if you find a nominally 6V robotics motor, it's probably safe to operate it on 5V.

Follow the generic wiring plan in Chapter 47, Feedback Device Wiring:

- Connect a diode between the (+) and (-) leads of the motor, as described in Chapter 53, Coil Diodes. Diodes are always required for motors because of the magnetic fields created by the motor's operation.
- Connect the (+) terminal of the motor to the positive voltage from your power supply
- Connect the (-) terminal of the motor to an available output port on your output controller

If the motor's terminals aren't labeled (+) and (-), it probably doesn't matter which order the wires are connected. Most DC motors will simply run backwards if the polarity of the power connection is reversed. Gear motors sometimes have a preferred direction because of the gearing, though, so do check markings for a preferred polarity. If the motor terminals *aren't* marked, you might want to try both polarities to see if one produces a better sound effect, since the gears might sound

rather different running in each direction.

Electrical interference: Be sure to use a diode with the motor. If you still have electrical interference problems with the motor (for example, USB devices randomly disconnect, or you see random keyboard input on the PC), you might need to add more filtering. The two-inductor filter described for the shaker should work equally well for a gear motor. See "Electrical Interference" in Chapter 61, Shaker motors for the wiring and parts details.

DOF setup: In the DOF Config Tool, go to your Port Assignments page. Find the output controller port number where you wired the motor. Assign this to "Gear" (which is short for "Gear motor").

63. Fans



At least two real pinball machines featured fans mounted on top of the backbox: *Whirlwind* (Williams, 1990), *Twister* (Sega, 1996). Two titles might seem like an awfully skimpy reason to include a fan on a virtual cab, yet this has become one of the more popular toys, and I'd highly recommend it. The reason? Fans are uniquely tactile. They add a completely new sensory effect to the playing experience.

As you'd expect, the DOF database replicates the authentic fan action in *Whirlwind* and *Twister*. Including a fan on your cab will make those games that much closer to the original. Now, if that were all the use you'd get out of a fan, it probably wouldn't be worth the trouble unless you had a rather fanatical devotion to one of those specific games. Fortunately, that's far from the only use your fan will see. The DOF database is programmed to trigger the fan in dozens of games - 41 as of this writing.

I really like the fan effect and the way DOF uses it, and I think most pin cab builders who've included them would agree. I rank this toy near the top of my priority list. It's a lot like the shaker motor in terms of its effect on game play: it just makes the game feel more exciting and immersive.

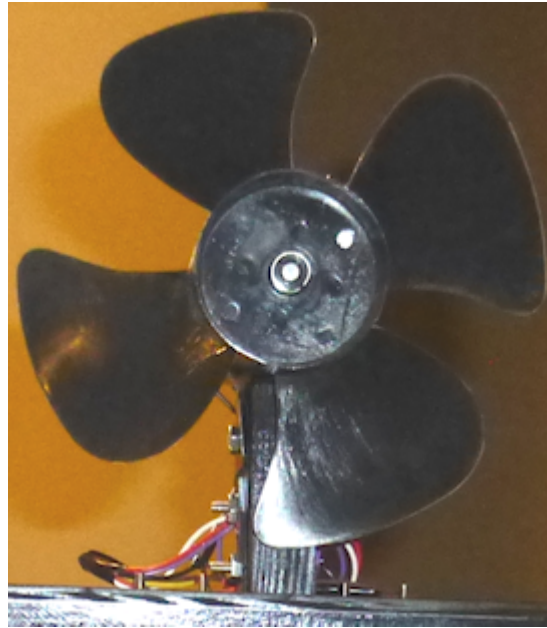
Auto/boat fans

The easiest way to add a fan is to use an off-the-shelf fan made for use in cars or boats. These can be found in the right size range to fit nicely on top of a backbox (about 6" diameter), and anything made for a car or boat will run on 12V DC, which means you can power it from a secondary ATX power supply. Search on Amazon for "car boat fan" to find options.

DIY

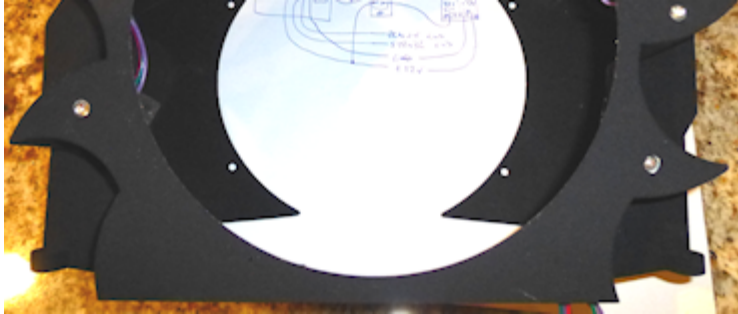
For my own cabinet, I wanted something that looked more custom, to fit with my cab's artwork, analogous to how *Whirlwind*'s fan enclosure is shaped like a storm cloud. I started with a bare fan assembly, just the fan blade and motor, and built my

own custom 3D-printed plastic enclosure around it.



My DIY fan assembly. A run-of-the-mill 12VDC motor with a 1/4" shaft, with a 6" plastic fan blade press-fitted onto the shaft, mounted on a makeshift wood and sheet metal bracket. The blade is the same one that Williams used for Whirlwind, which happens to be an OEM part for microwave ovens (!).





My 3D-printed fan enclosure (in the wiring stages, installing LEDs around the perimeter).



The fully assembled Pinscape fan.

If you want to go the DIY route, there are a couple of ways to get a bare fan assembly to use as the core. One is to buy a car/boat fan like above and remove its case. Another is to build your own entirely DIY fan assembly from parts.

The completely DIY approach is probably easier. It's certainly more predictable; you never know with a pre-assembled fan if it'll be possible to remove the case without destroying the whole thing. To build a fan from parts, you really only need two pieces, both of which are easy to find:

- A 12V DC motor with a 1/4" shaft. You don't need anything particularly special; most 12V motors will have plenty of power to run a fan. Look on eBay, Amazon, or robotics hobby vendors.
- A blade. The original Whirlwind fan, amusingly enough, was actually an OEM part for microwave ovens (Thorgren model number 6C2504C1). It's plastic, has four blades, and a 1/4" bore, like the one pictured at right. You might be able to find that exact part on eBay or appliance parts vendors, but there's really no need unless you have a special fondness for *Whirlwind* in particular. You can find any number of similar appliance fan blades on eBay by searching for **6" fan blade**. Look for blades with a 1/4" bore, as these will easily press-fit onto a 1/4" motor shaft.



You'll also need to build a bracket to attach the motor to the backbox. You should be able to improvise something if you have some minimal woodworking skills. (You'd need to build this part even if you're re-purposing an auto/boat fan, so I didn't count it in the "two pieces" I said you needed for a fully DIY fan.)

In-cab blowers

A few pin cab builders have built fans into the cabinet body rather than putting it on the backbox. One technique mentioned on the forums was to use an air duct to to route air from a small fan through an opening in the coin door. (One potential opening to use for this: the dollar acceptor slot. Newer coin doors made for the US market tend to include a large square opening intended for dollar bill acceptors.)

Here's a particularly amusing take on this theme:

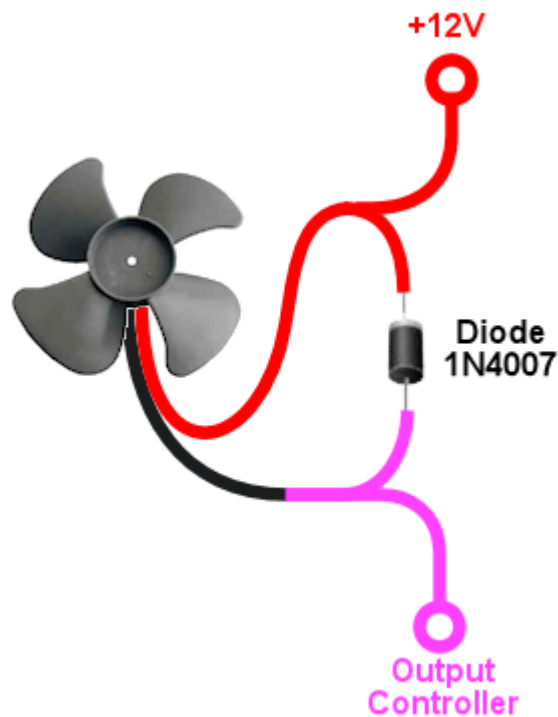
Heavily modified whirlwind topper fan (Pinside pinball forums)

I personally like the effect of the fan blowing at you from the backbox, but at least one person who built the coin door version commented on the surprise value of the sneaky placement.

Wiring a fan (DC voltage)

Assuming you're using a 12V DC fan or motor, you can power the fan from the +12V terminal (yellow wire) of your secondary ATX power supply. See Chapter 45, Power Supplies for Feedback.

Connecting the fan is just like any other feedback device:



Connect one terminal of the fan motor to the +12V from your secondary ATX power supply. Connect the other terminal to an available port on your output controller.

The diode is required to protect your output controller and computer electronics from interference from the magnetic field generated by the motor. See Chapter 53, Coil Diodes.

If you're using an LedWiz, don't connect the fan directly to the LedWiz. Fans use too much power for an LedWiz, so you'll need some kind of booster circuit. If you're using the Pinscape expansion boards, you can connect the fan directly to any MOSFET Power Board port.

Electrical interference

Be sure to use a diode with the motor, as shown above. If you still get electrical interference when it runs (for example, USB devices randomly disconnect, or you see random keyboard input on the PC), you might need to add more filtering. The two-inductor filter described for the shaker should work equally well with a fan motor. See "Electrical Interference" in Chapter 61, Shaker motors for the wiring and parts details.

Wiring a fan (AC voltage)

If you're using an AC-powered fan, **don't** connect it directly to an LedWiz, Pinscape controller, or any other solid-state controller. You'll need a relay for this instead. If you're using a Sainsmart relay board, the outputs on those are in fact relays, so you can connect an AC device directly. For LedWiz and Pinscape, you'll need to add a relay to the circuit: your output controller will control the relay coil, and the relay will control the AC motor. See "AC devices" in Chapter 47, Feedback Device Wiring for the wiring plan.

DOF Setup

In the DOF Config Tool, go to the Port Assignments page. Find the port number where you wired the fan. Assign it to "Fan".

At the right side of the page, you'll also find a section labeled "Fan" that lets you set the intensity range. If you're using a PWM-capable controller (e.g., an LedWiz or a Pinscape power board), this lets you set the range of power that DOF uses when the fan runs. The intensity values are on a 1-48 scale, where 48 is the highest power. The default settings use the full available range. If you find that the fan is too powerful (runs too fast) when DOF activates it during game play, you can reduce the maximum intensity setting to slow it down. Similarly, if the fan seems too weak some of the time, you can raise the minimum setting.

64. Chimes and Bells

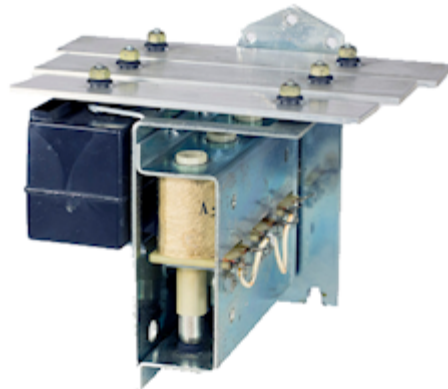
Next time you're watching a TV show or movie, and you notice a pinball machine being played in the background, pay attention to the sound effects. I can almost guarantee you that the sound effects will mimic the bells and chimes from a 1960s electromechanical machine, even if the on-screen machine is a 1980s or 1990s solid-state table. You and I know that those solid-state machines actually have electronic sound systems that include speech and music, but when you see them in TV and movies, the sound effects will almost always be EM chimes. Why is this weird technical gaffe so comically predictable? Maybe in part to avoid music licensing costs, but the main reason, I think, is that the sound designers know that those chimes are what most people think of when think of pinball. They're not going for technical accuracy with those sound effects; they're trying to create a sonic impression. EM chimes are the "audio icon" for pinball.

Which brings us to the topic at hand: chimes and bells in a virtual cab as physical feedback devices. Before the solid state machines came along in the 1980s, pinball machines made those iconic bings and bongs with little solenoid-driven percussion instruments. These instruments came in two main types. The first is known as a "chime unit", which is essentially a miniature xylophone consisting of three or four metal keys. The second type is bells, typically the circular half-shell type you might find in an old-fashioned alarm clock or fire alarm.

If you play only games from the 1980s and later, these will be of little interest to you. If you enjoy games from the electromechanical era, though, real chimes and bells can be a great enhancement. Recorded effects played through speakers can't approach the visceral impact of real percussion instruments.

Chime units

A chime unit is a miniature xylophone played by solenoid-actuated hammers. The xylophone keys are metal bars tuned to selected musical notes, and each key has a solenoid positioned underneath with its plunger aimed at the key. When the solenoid fires, the plunger pops up and strikes the key, setting it ringing.



The chime unit pictured at right is a classic Gottlieb model, which was mass-produced and used in numerous Gottlieb EM titles throughout the 1960s and 70s. The black box at the back is a resonance chamber: an enclosed air space sized to resonate at each bar's fundamental frequency, to amplify the sound and make the ringing last longer. The resonance chamber is probably the most important part of the design acoustically, since it can make the notes sound richer and warmer by sustaining the ringing effect. Williams and Bally also manufactured chime units with similar basic designs, but Gottlieb's is widely considered to be the most sonorous, thanks to its particularly well-designed resonance box.

What to buy: The pinball manufacturers had a few standard chime unit designs that they mass-produced during the 1960s and 1970s. They stopped making the originals a long time ago, but you can often find used ones on eBay. There are also two modern reproduction models available for purchase (as of March 2022):

- Marco Specialties sells a 3-bar reproduction based on the Gottlieb units from the 1970s. Search for part number B-12023. About \$300.

- McCullough's Chime Unit, available at www.chimeunit.com. This is apparently an approximation of the original designs, not an exact replica of any particular original. I haven't seen these in person; the reviews I've read are positive but say that they don't sound quite as good as the original Gottlieb units. About \$150.

DIY chime unit

Given the difficulty of finding an original chime unit and the high prices of the modern reproduction units, a few pin cab builders have at least explored the idea of building their own chimes from scratch. The basic design is straightforward, but the details are tricky enough that I don't think many people have successfully built them. I haven't attempted it myself, but it seems like an interesting project, so I'll pass along a few things I've learned from reading about other people's attempts. If you do build your own, I'd love to hear about how you went about it and how it turned out.

The basic components are:

- Metal bars to serve as the xylophone keys
- Solenoid hammers to strike the bars
- A resonance chamber to sustain the ringing effect

The solenoid coils are easy to find. You need nylon-tipped plungers for the coils to serve as the hammers; the ones used in replay knockers should work. You could just buy a replay knocker coil and plunger for each key, although knocker coils are probably overpowered; the original chime units had smaller 24V coils.

You can find replacement metal bars for the original chime units from online pinball vendors. It's also fairly easy to create your own from raw metal stock, which you can find at hardware stores or online. You just need to cut bars to appropriate lengths for the different chime tones. There isn't a simple formula for length vs. frequency, since many variables go into it; it depends on the type of metal as well as the thickness and length of the bar. Other things being equal, though, a longer bar produces a lower tone. You can find Web sites on the subject if you want to work out the math, but I think it might be easier to do some ad hoc experiments by cutting metal bars to different lengths until you find a pleasing set of tones.

Some additional tips:

- Use a solenoid plunger with a nylon tip, not a metal tip. A metal tip will make a sharp metal-on-metal sound; a nylon tip will just set the bar ringing, producing the bar's pure tone without any sharp contact noise.
- Support the bars with small rubber grommets, to minimize damping (you want the bar to ring after being struck). You can find purpose-made grommets as replacement parts from pinball vendors, but generic grommets from a hardware store will probably work just as well.
- Place the supports at two points, centered relative to the bar's width and inset from each end by about 22% of the total bar length. To find the exact positions, sprinkle some sawdust or salt evenly over the bar and give it a whack with a hammer. The dust/salt will settle at the nodes of the bar's acoustic vibration waveform, which are the exact points where the supports should go because they're stationary while the bar is ringing.

More on the acoustics of xylophone-type chimes: www.mmdigest.com/Gallery/Tech/XyloBars.html.

Aside from the xylophone keys, a good chime unit also needs a resonance chamber: a rigid box with an opening on the side facing the chime. This serves to sustain the ringing sound for a time after each hammer strike, which creates a more sonorous effect than the chimes alone. The size of the box has to be chosen according to the frequency of the adjacent chime, so each bar needs its own box. You can find formulas for calculating the resonant frequencies of chambers with various shapes in the Wikipedia article on acoustic resonance.

The resonator boxes in the original chime units were made of some kind of hard plastic. I don't think the material is critical, as long as it's something rigid and fairly smooth, so that it doesn't absorb much of the sound energy. I imagine that you could make this work with 3D-printed plastic.

Here's a forum post about a DIY chime unit (using metal pipes rather than the more conventional bars):

Homemade Tubular Pinball Chimes

How many chimes?

Gottlieb and Bally each mass-produced their own three-chime units in the 1960s and 70s, and used them as standard parts in most of their games during that period. That makes the three-chime setup by far the most common in EM games you're likely to see re-created in the virtual systems.

In addition, Bally created a four-chime system that they used in a few machines from about 1976 to 1979. This wasn't actually a whole new design; it consisted of their pre-existing three-chime unit, plus a second unit with a single, extra-long chime bar installed. The combination extended the three-tone set to four tones, with the fourth tone being a new extra-low note.

DOF has a provision for a fifth chime bar as well, but as far as I can tell from reviewing IPDB game descriptions and other sources, there were never any real machines that actually used a fifth chime. DOF's use of it appears to be entirely in the "fantasy" realm - an embellishment for virtual pin cabs, not something that replicates a real machine feature. So if you do install a fifth chime, it's completely up to you what tone it plays: there's no authentic reference point from the real machines for what it "should" sound like.

For a pin cab, I'd personally go with a three-chime unit. Only a few real games were made with the four-chime units, so the extra work and cost of a fourth physical chime probably isn't worth it to most people. And DOF's fifth chime slot is purely a DOF embellishment, and is only used by a handful of tables. You can always map DOF's fourth and fifth chimes to fire your other physical chimes, so you'll still be able to get something to happen when it fires, even though it won't be a distinct tone.

Musical notes

The real three-chime units typically play notes in the fifth octave, usually about two musical notes apart. If you're designing your own chimes, a good target set of notes might be C5, F5, A5. The extra chime on the Bally four-chime units is reportedly about a C4.

Shell Bells

Chime units weren't the only type of percussion instrument in the EM machines. Some machines had "shell" bells instead: circular shell bells, with solenoids positioned to strike them at the edge. These bells are similar to those you'd find on a

children's bicycle.

Machines equipped with bells usually had two of them, one larger and one smaller, to produce different tones. A common arrangement was one 3" bell and one 5" bell.

You can buy similar bells as replacement parts from the pinball supply vendors; look for "bell" and "bell assembly". You can also find these used on eBay.



You could also use any sort of similar bell, such as a bicycle bell, a bell from an old alarm clock or phone, from a fire or burglar alarm, etc. You can improvise your own firing mechanism using a pinball bell strike coil assembly, or using a cheap eBay push-type solenoid.

Chimes and bells were mutually exclusive in the real machines, as far as I've seen. They serve exactly the same purpose and worked the same way, the only difference being the shape of the ringing element. The pinball manufacturers probably chose whether to use chimes or bells in a given machine according to parts cost and availability, and perhaps the whims of the game designers.

For a virtual cabinet, should you install bells, chimes, or both? In my opinion, a chime unit should be your first choice. Chimes were by far the most common noisemaker in the real EM machines over the years, so a chime unit will create the most authentic simulation for most games; and a chime unit can easily stand in for shell bells for games that used those instead. My second choice would be a chime unit *plus* a pair of shell bells (one large and one small), if your budget allows it and you have space in your cabinet. Shell bells do have a different tonal quality compared to chimes, so they'll add some variety to your EM games, and they'll give you better authenticity for games that used them.

If you do install shell bells, you can assign them to the **Shell Bell Large** and **Shell Bell Small** devices in the DOF Config Tool.

Repeating bells

This isn't exactly in the same category as chimes, but it's related. A few real machines over the years have featured repeating bells, like the ringer in an old-fashioned telephone, or a fire alarm. The key is that the hammer rapidly hits the bell over and over as long as the bell is energized, rather than just hitting it once.

Space Shuttle (Williams, 1984) has a bell like this that fires each time you complete the stand-up targets, and also serves as the replay knocker. *Taxi* (Williams, 1988) also has a bell like this, which sounds when multiball starts or when you collect a jackpot.

You might be able to find the original bell unit for a game that featured one (such as *Space Shuttle* or *Taxi*) from a pinball parts vendor. It might be easier to find some other electrically operated bell and adapt it, though. Look for "12V alarm bell" on eBay, for example.

The DOF Config Tool has a device type called **Repeating Bell** that you can assign to a physical device like this, if you install one.

Bells as toppers

Large decorative bells make good backbox toppers. Some real machines used bell toppers as part of their theming, most famously *Fire!* (Williams 1987).

You might be able to find the specific replacement part for *Fire!* from a pinball vendor or used on eBay, but it's probably easier to find a random decorative bell with the right size and shape. As with the shell bells, you can improvise your own firing mechanism using a pinball bell striker coil assembly, or a cheap eBay push-type solenoid.



If you assign a decorative bell, you can assign it to the device named **Bell** in the DOF Config Tool.

Installing in a virtual cab

If you found a chime unit (or built your own), it will probably be too big to fit in the backbox. In the original EM machines, these were typically mounted in the front corner of the main cabinet, under the plunger. I think they mounted them there simply because it was a convenient place to put them, not because it's a special spot in terms of acoustics, so install it wherever you have space available.

Bells were more typically mounted in the backbox. Again, though, I don't think there's any need to slavishly replicate this placement. Placing them in the main cabinet should also work if you don't have room in the backbox.

Wiring

Bells and chimes are wired just like any other output device, as described in Chapter 47, Feedback Device Wiring.

Each chime in a chime unit has its own coil, so simply wire each coil to a separate output controller port.

If you're using the Pinscape expansion boards, you can connect a chime or bell directly to any MOSFET Power Board or Chime Board port. If you're using an LedWiz, don't connect a chime coil directly to it, since these coils use more power than the LedWiz can handle; you'll need some kind of booster circuit to use chime with an LedWiz.

Always use diodes with coils. Diodes are required to protect your output controller and other electronics from interference from the magnetic field generated by a coil. See Chapter 53, Coil Diodes.

If you're using a repeating bell (the type where the hammer strikes repeatedly), it might cause more electrical interference than a regular coil does due to the way it operates. A ringing bell uses a mechanical switch in the hammer to produce the rapid repeat firing, by interrupting the power to the coil every time the hammer strikes. This mechanical switching action can inject a great deal of electrical noise into the circuit. If you have any problems after installing the bell with your other electronics (e.g., the output controller resets or disconnects from USB when the bell fires, or other devices fire when the bell fires), try adding a small capacitor (maybe a 0.1uF disc capacitor) in parallel with the coil diode. If possible, place it directly across the mechanical switch terminals inside the bell; if you can't do that, place it physically close to the bell's electrical terminals.

Timer protection circuits

Pinball coils like those used in bells are designed to be fired in short bursts only.

They'll overheat if they're energized for long periods. Under normal conditions, this isn't a concern for chime coils, since games that use chimes should naturally fire them in short bursts, exactly as they're designed to be used. However, PC software sometimes has bugs or glitches, and one of the things that can happen when something goes wrong on the PC is that an output port can get stuck on. This can destroy a pinball coil if it's not quickly fixed.

The Chime Board from the Pinscape expansion boards has a dedicated hardware timer for each output to protect against exactly this failure mode. If you're using the Pinscape boards, it's recommended to use Chime Board outputs for chimes and bells.

Note that this doesn't apply to repeating alarm-type bells, since those are intended to fire for a longer periods.

If you're using the Pinscape software but not the Chime Boards, you can still get some of the same protection by using the "Flipper Logic" feature for your chime ports.

See Chapter 54, Coil Timers for more details.

DOF Setup

In the DOF Config Tool, go to the Port Assignments page. Find the port numbers where you wired your chime/bell coils. Assign to the appropriate DOF devices:

- **Chime Unit High Tone** through **Chime Unit Low Tone** are for traditional three-bar chime units. Assign the shortest bar to **Chime Unit High Tone**, since the short bar has the highest-pitched tone. Assign the remaining bars in descending tone order.
- **Chime Unit Extra-Low Tone** is for the fourth chime in the Bally four-bar chime units from the late 1970s. If you have a fourth chime bar, assign the **Extra-Low Tone** DOF slot to the physical bar that produces the deepest tone (the one with the longest bar). If you have a more typical three-chime unit, you can assign this DOF slot to the same physical chime as your "Low Tone" port.
- **Chime 5** is an extra DOF chime that doesn't correspond to any physical chimes from any real machines, but is used as a DOF embellishment for a few tables, to add some extra variety to their sound effects. If you have a fifth physical chime device, you can assign it here. If not, you can assign this to the same physical device as one of your other chimes. Very few games use this, so you won't miss much if you just leave it unassigned.
- **Shell Bell Small** and **Shell Bell Large** are for shell bells like those found in some 1960s machines (see shell bells above). These work exactly like chimes, but use circular metal bells rather than bars as the ringing element, for a different tonal quality.

If you install physical shell bell devices, assign them to these DOF slot. If you don't have physical shell bells but you do have a chime unit, assign **Shell Bell Small** to your shortest chime bar, and assign **Shell Bell Large** to your longest chime bar.

- **Repeating Bell** is for a bell that rings continuously when energized, like on an old-fashioned telephone (see repeating bell above). If you have such a device, assign it to this slot. Since this slot is specifically for repeating bells, I wouldn't map it to any other single-firing chimes or bells; I'd just rely on the digital sound effects recordings in the games instead for this one.

- **Bell** is intended for a large novelty bell of the sort found as the backbox topper on *Fire!* If you have a device like this, assign it here. If not, and you have chimes or shell bells, I'd assign this slot to fire the one with the deepest tone.

65. Addressable Light Strips

Addressable or "smart" light strips are similar to the basic light strips most cab builders use for Chapter 58, undercab lighting, but with the added feature that each LED on a smart strip can be set to a different color. That's what "addressable" means: each LED can be addressed - commanded - individually.

In pin cabs, these can be used in a variety of ways, such as:

- A strip of lights down the entire length of each side of the playfield TV, usually mounted on the side wall of the cabinet just above the TV. These can show chaser lights that follow the ball at certain times (such as when launching from the plunger), and can also show flashes of light situated near playfield flasher lights on the TV, to reinforce the video rendition of the flasher with a brighter nearby LED flash. (See Chapter 56, Flashers and Strokes.)
- One strip or several rows of strips across the width of the back edge of the playfield TV. This can display more ball chaser light effects, and can also take the place of a dedicated flasher panel (see Chapter 56, Flashers and Strokes). In matrix form (multiple rows of strips), it can also serve as a sort of supplemental DMD, to display additional dot animation effects during game play.
- A strip or several rows of strips mounted near the top of the backbox. This can simulate "beacon" devices (see Chapter 57, Beacons).
- Addressables can also be used to illuminate the flipper and MagnaSave buttons (see Chapter 55, Button Lamps). This function can also be performed (more easily, in my opinion) by a general-purpose output controller like an LedWiz or Pinscape unit, but you could use addressables if your system doesn't include a general-purpose controller at all, or if all of its ports are committed elsewhere.

The placements listed above are the most common, and DOF has pre-programmed effects for them. But you're not limited to those setups. The DOF Config tool lets you create your own custom programming in addition to the pre-programmed effects. So if you have other ideas about how to arrange the strips, you're free to set them up as you like, as long as you're willing to do the additional work to program the effects yourself.

Controller

You **can't** use an LedWiz, Pinscape, or other general-purpose output controller to run a smart light strip. Instead, you need a separate, special type of controller just for these strips. That makes smart strips different from all of the other output devices we've looked at.

The required controller is a bit of a DIY project. Being a DIY project, you can build the whole thing from scratch any way you like, but fortunately there's a standard recipe to follow. If you want to use the standard setup, here are the parts you need:

- Teensy 3.2 (an Arduino-like microcontroller board, about \$20)
- OctoWS2811 adapter for Teensy 3.2 (interfaces the Teensy to CAT6 jacks for easier wiring to the light strips, about \$10)
- A CAT6 cable to connect the light strips to the adapter
- A USB micro to USB "A" cable to connect the Teensy to the PC

The Teensy isn't a light strip controller by itself. It's just a microcontroller (a small single-board computer). To turn it into a light strip controller, you need some special

software:

TeensyStrip Controller (GitHub)

Follow the instructions there to download the software onto your Teensy. The project Wiki linked there also has more information about setting up your light strip hardware, so you might look there if you run into any issues not covered in this chapter.

Parts for the light strips

Look for **WS2812** light strips on eBay.

The strips are available with 30, 60, and 144 LEDs per meter. As you'd expect, the higher densities are more expensive. For strips along the side of the cabinet, I'd recommend the 60/meter type. 60/meter is also good for a single strip across the back. If you're attempting to create a DMD-like effect by using multiple rows at the back, though, you'd probably prefer the 144/meter type there, as the higher density will allow for better rendering of shapes across multiple "pixels".

WS2812B strips are equivalent to WS2812. (The difference with the "B" type is related to the physical solder pad geometry of the individual LEDs. That makes no difference when you're buying a strip, since the LEDs are already soldered to the strip. It would only matter if you were buying the individual LEDs to make your own custom strip or circuit board layout.)

You can also use WS2811 light strips, but I'd stay away from those because they're less predictable. In particular, some of the WS2811 strips are set up so that the LEDs can only be controlled in groups of 3, not individually. The WS2812 LEDs are always individually addressable, which is what you probably want if you're going to all this trouble. If you do buy a WS2811 strip, be sure to check the seller's description carefully to make sure that the elements are individually addressable.

Power supply

The strips require a 5V power supply. You can use your secondary ATX power supply if you have one, or a dedicated 5V DC supply. See Chapter 45, Power Supplies for Feedback.

Make sure that your power supply has enough capacity. Long light strips can consume a surprising amount of power. Computer ATX power supplies are usually sufficient for typical pin cab light strips, but smaller "wall wart" power supplies might not have enough power for long strips.

To determine the total power requirement, count up all of the individual LEDs across all of your strips, and multiply the total number by 60mA. This will give you the total current used. Many power supplies list their capacity in Watts rather than Amps, though, so you might have to convert. For a 5V supply, use this formula:

$$\text{Watts} = \text{Amps} \times 5$$

Note that this formula is written in terms of Amps, not milliamps. To convert from mA to Amps, simply divide by 1000.

For example, if you have two meters of 60 LED per meter strips, you have 120 LEDs. $120 \times 60\text{mA} = 7200\text{mA} = 7.2\text{A}$, or 36 Watts (7.2×5).

Once you determine the power requirements for your LED strips, check to make sure

that your power supply is rated to deliver at least that much power. A wall wart supply or power brick usually has markings showing the rated Amps or Watts.

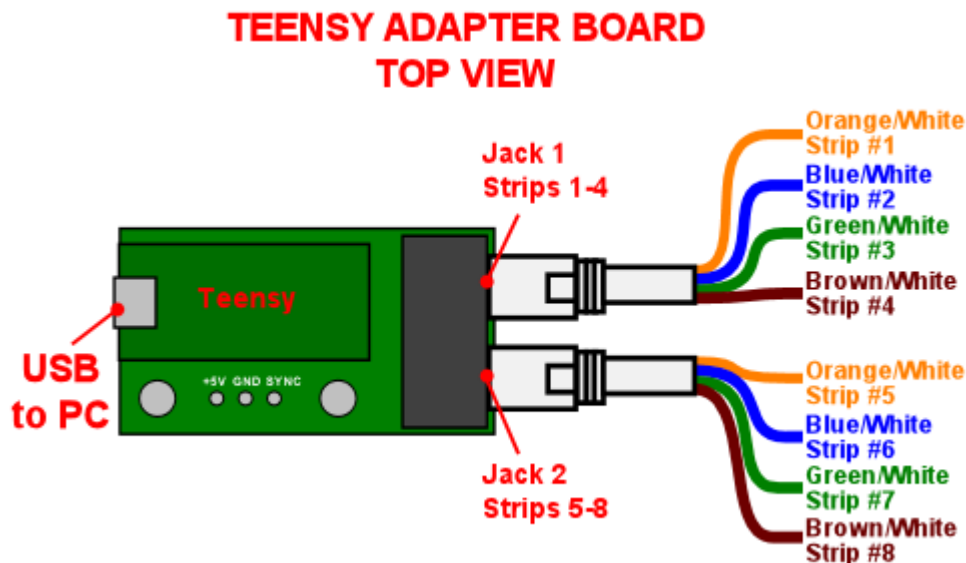
Wiring

Two connections to the LED strip are required: the power supply, and the control signal from the Teensy.

You only need a single control signal connection to each contiguous strip. This always connects to the very end of the strip. The strips have a directional signal flow, with arrows marked on the strip to show the signal direction. Your data connection has to go at the "starting" end of the strip, so that the arrows point away from that side.

If you're using a CAT6 cable, prepare it to connect to the strip like this:

- Cut off the plug from one end, as you need bare wires to attach to the strips
- Strip off at least a foot of the **outer** insulation jacket, being careful not to cut any of the wires inside. Strip more if you need to in order to make the wiring reach multiple strips.
- Carefully unwind the four twisted pairs from each other, keeping the individual pairs twisted together.
- Each twisted pair connects to one light strip. See the diagram below.



Note! The wire colors are for CAT6 cables using T568B termination, which is how most cables are set up. However, some cables use T568A termination, which looks the same but swaps the orange and green pairs.

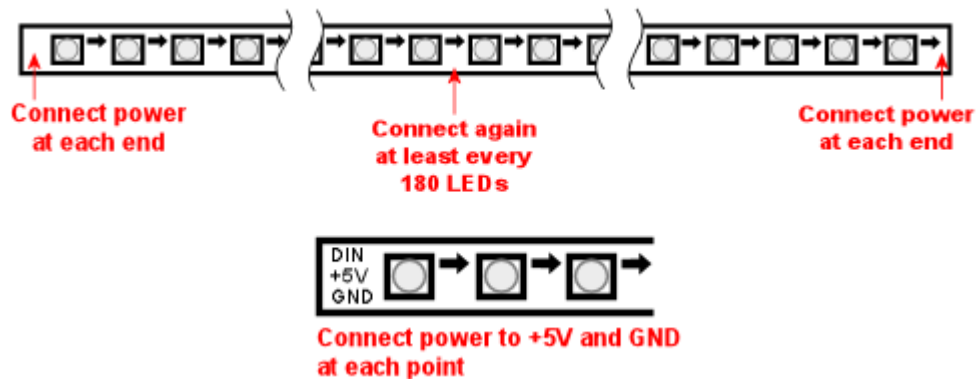
The connection from the controller uses the two wires from a twisted pair from the CAT6 cable. Each twisted pair consists of a solid color wire and a white wire striped with the same solid color. For example, a solid orange wire is paired with a white/orange striped wire. Connect the solid color wire to the **DIN** pad on the strip, and connect the striped wire to **GND**.



**Data connection at this end
(observe arrows printed on strip)**

Connect to DIN & GND pads

For power, you must provide a separate connection at each end of the strip, and an extra connection **at least every 180 LEDs**. The extra connections are required because the copper traces on the strips are limited in the amount of current they can carry, and every LED between power connections adds to the current used. Multiple connections are required to distribute the load so that the traces don't carry too much current through a single section.

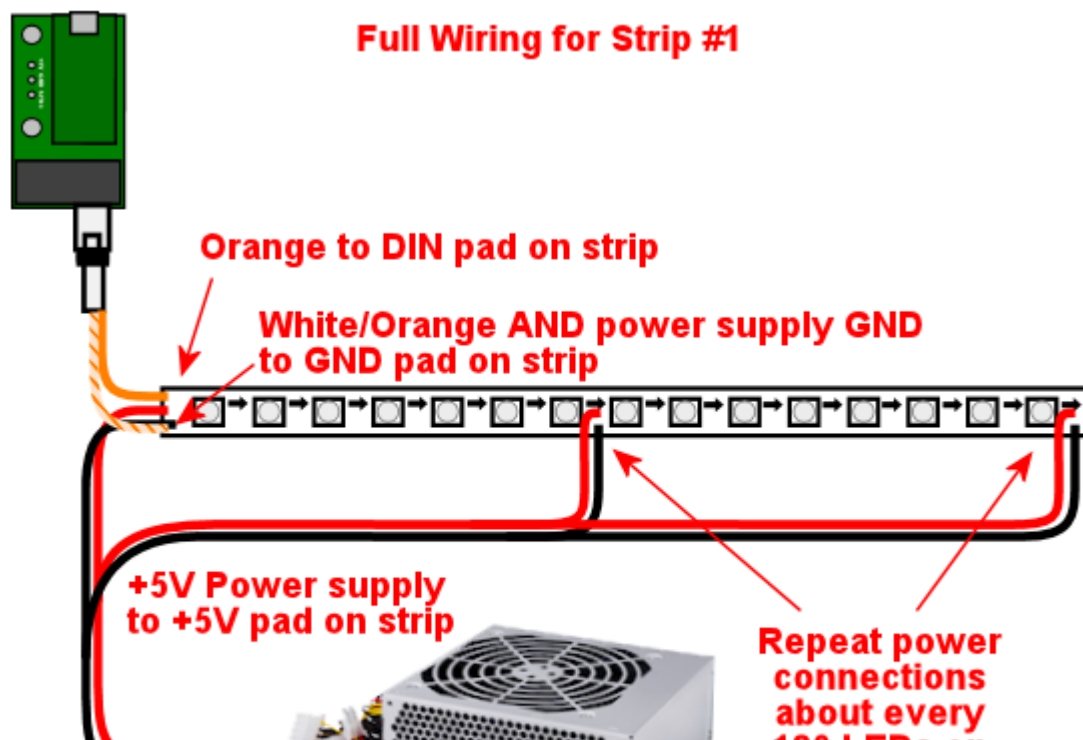


At each point where you connect power, connect the power supply's 5V terminal (the red wire on an ATX power supply) to the **+5V** pad on the strip, and connect the power supply 0V/ground (black wire on an ATX supply) to the **GND** pad on the strip.

For power connections in the middle of a strip, leave the DIN/DOUT pads unconnected at that point. You only need the single DIN connection at the "input" end of the strip.

For the power connection at the start of the strip, where you're also attaching the data connection, note that the GND pad on the strip will connect to **both** the power supply 0V/ground and the white striped wire from the CAT6 twisted pair.

Here's a full wiring diagram for the first strip. Each additional strip is wired the same way, but connects to a different wire pair from the CAT6 cable: the second strip connects to the blue/white pair, etc.





Connecting multiple strips

You'll probably have several sections of strips, in which case each strip needs its own data input connection. There are two ways to handle multiple strips:

- Connect each strip to a separate twisted pair in the CAT6 cables from the Teensy adapter. The adapter has plugs for two CAT6 cables, and each cable has four twisted pairs, so you can connect up to eight strips this way. See the diagram of the adapter above to figure out which wires in the CAT6 cable connect to which strips.
- Daisy-chain the strips. Using twisted pair wiring, connect wires from the **DOUT/GND** pads at the **end** of the first strip to the **DIN/GND** pads at the start of the second strip. Repeat for each additional strip.

Most people find the first approach (wiring each strip directly to the Teensy adapter) to be easier and more reliable. Daisy-chaining is possible, but you have to provide good clean connections between the strip segments, using twisted-pair wires, to make it work. The challenge is that the data signal operates at high frequencies and can be very sensitive to electromagnetic interference from other devices in the cab. Twisted pair wiring provides a degree of shielding.

Daisy-chained connections are more or less required, though, if you're creating a matrix of strips with multiple rows. In that case you'll probably have too many strips to use the direct connect approach.

Mounting in the cab

Most of the WS2812 strips are sold as bare strips without any adhesive backing, so you'll have to provide your own adhesive. Most people use double-sided foam tape.

If you're installing strips along the inside walls of your cabinet adjacent to the playfield TV, pay attention to clearance so that you don't make it impossible to get the TV in and out of the cab. Many people build their cabs so that the TV is a very tight fit, so strips that intrude even a couple of millimeters could make it difficult or impossible to lift the TV or remove it. If the strips are permanently installed along the inside cabinet walls, you'll be stuck without access to the inside of the cab, which you should never let happen. Here are a couple of approaches other cab builders have used:

- Mount the strips on a removable platform, such as a thin aluminum bar. Attach that to the cab wall with Velcro. This makes it easy to remove the strips to get them out of the way any time you need to lift or remove the TV. This is the way that many pinball collectors do it when they add similar light strips as mods to their real machines.

If you're using anything metallic as the removable platform, be sure to place an insulating layer between the strips and the metal. Many types of LED strips have exposed copper pads on the back, so they'll short out if you mount them directly to a metal surface. Foam tape is a good solution, because it can serve the dual purposes of sticking the strips to the metal and insulating the backing.

- Recess the strips into the wall, so that they don't get in the way of the TV. Use

a router to cut channels into the cabinet wall where the LEDs will mount. Make the channels deep enough that the LEDs are fully recessed, so that the front surface of the LEDs is flush with the interior wall.

Available DOF effects

The DOF Config Tool provides pre-programmed effects for the standard light strip placements. The easiest way to set up strips with DOF is to use these programmed effects. Here's a list of the available effects and how they're typically assigned to physical light strips in the cab.

DOF Effect Name	Use with	Description
PF Left Flashers MX	Left playfield TV strip	Simulates playfield flashers near the left edge of the playfield
PF Left Effects MX	Left playfield TV strip	Special effects along the left side of the playfield, such as ball chaser lights
PF Back Flashers MX	Playfield TV rear strip/array	Simulates a dedicated 5-flasher panel (see Chapter 56, Flashers and Strobes)
PF Back Effects MX	Playfield TV rear strip/array	Special effects near the rear of the playfield
PF Back Strobe MX	Playfield TV rear strip/array	Simulates a dedicated strobe light (see Chapter 56, Flashers and Strobes)
PF Back Beacon MX	Playfield TV rear strip/array or backbox strip/array	Simulates a dedicated
PF Back PBX MX	Playfield TV rear strip/array	Additional special effects used in PinballX
PF Right Flashers MX	Right playfield TV strip	Simulates playfield flashers near the right edge of the playfield
PF Right Effects MX	Right playfield TV strip	Special effects along the right side of the playfield, such as ball chaser lights
Flipper Button MX	Flipper button lamps	Illuminates the flipper buttons in the appropriate color for each game
Flipper Button PBX MX	Flipper button lamps	Additional flipper button lighting for PinballX
Left MagnaSave MX	Left MagnaSave button lamp	Illuminates the left MagnaSave button in the appropriate color for each game

Right MagnaSave MX	Right MagnaSave button lamp	Illuminates the right MagnaSave button in the appropriate color for each game
RGB Undercab Complex MX	Undercab lights	Ambient illumination effects for undercab lighting (see Chapter 58, Undercab Lighting)

DOF Setup

There are two parts required to set this up with DOF: a "cabinet configuration" file, and the DOF Config Tool settings.

Part I: Cabinet config file. You'll have to manually create a file on your PC called `Cabinet.xml`, in the DOF folder, to describe your hardware setup for the strips.

Before you do that, though, you *also* have to set up a "global" config file to tell DOF to use your `Cabinet.xml` config file. (Nothing's ever easy with DOF!) That procedure is explained in "Extra controller setup" in Chapter 46, DOF Setup. Please read through that section and follow the steps listed there. That will give you a starting point for the `Cabinet.xml` file that you can fill in with the light strip information.

The light strip entries in `Cabinet.xml` file are quite complex, and they're covered in the Wiki page for Swiss Lizard's Teensy code, so I'm not going to reiterate all of that here. If I copied it here, it would just gradually drift out of date and become more confusing than helpful. Better to go straight to the source:

github.com/DirectOutput/TeensyStripController/wiki - see "Software: DirectOutput Framework"

Another helpful resource is this VPForums thread, which has examples of the cabinet config file.

How to Set Up Addressable LED Strips (at VPForums)

Part II: DOF Config Tool settings. Once you have the `Cabinet.xml` file set up, DOF will be able to find your hardware. But wait! There's more! You have to go through yet another procedure now to tell DOF to actually use that hardware.

This procedure uses the DOF Config Tool. Hopefully you're already familiar with that from setting up your general-purpose output controller. If not, please read through "The DOF Config Tool" in Chapter 46, DOF Setup.

The first step is to tell the Config Tool that you have a Teensy light strip controller device:

- Click the "My Account" tab to go to the account settings page
- Set "Number of WS2811 Devices" to 1. (This reflects the number of Teensy light strip controllers, **not** the number of light strips. If you have so many strips that you need two or more Teensy devices to control them all, set this accordingly.)

The second step is to create "combined effects". This is required because the Config Tool has multiple effects that *usually* end up being assigned to the same physical light strip. The reason for breaking these out as separate effects is that some people with very elaborate setups might actually have a separate physical light strip for each effect. But most people have simpler setups.

For example, the Config Tool has effects for "PF Left Flashers MX" (simulated playfield flashers along the left side) and "PF Left Effects MX" (other non-flasher effects for the left side). DOF separates these effects because you *could* provide two separate light strips for these effects. But most people don't; most people just run one strip up each side of the TV. If you're in the latter camp, you'll want to combine these into a single effect, so that you can assign that effect to your left strip:

- Click the "Combine Toys" tab
- In the "Toy Category" column, select RGB Addressable from the drop list
- In the "Toy 1" column, select "PF Left Flashers MX"
- In the "Toy 2" column, select "PF Left Effects MX"
- Click the "Add" button and confirm the change
- Repeat for each set of combined effects you'd like to create.
- Save changes

Here's a list of the typical effect combinations:

- PF Left Flashers MX + PF Left Effects MX (use for a single physical left playfield strip)
- PF Right Flashers MX + PF Right Effects MX (for a single physical right playfield strip)
- PF Back Flashers MX + PF Back Strobe MX + PF Back Effects MX + PF Back Beacon MX + PF Back PBX MX (for a physical rear playfield strip or array)
- Flipper Button MX + Flipper Button PBX MX (flipper button lamps)

The final step is to tell the Config Tool about your physical strips and what each one should be used for. The DOF Config Tool thinks of the individual strips as "ports" in your Teensy controller. This is analogous to the way the general-purpose output controllers work: in a regular controller, you tell DOF that "port 1 is my shaker motor, port 2 is the replay knocker..."

Now, with smart strips, it's a little different, because DOF doesn't think of smart strips in terms of physical device types like "left playfield strip" the way it does for "shaker motor" or "replay knocker". It's a little more abstract. The equivalent device types for smart strips are the *effects* listed under "Available DOF effects" above. Also, since smart strips are RGB devices, each physical strip takes up three DOF ports.

So putting those together, you set up your physical strips by saying things like "Port 1-3 is my PF Left Flashers MX strip". You can also use any "Combo" effects you set up above: "Port 4-6 is my Combo2 strip".

- Click the "Port Assignments" tab
- Select **WS2811 - directoutputconfigini30** from the **Device** drop list
- Set **Port 1** to the effect that you want to assign to your first light strip (the strip connected as "Strip #1", the orange/white wire pair from the wiring diagram earlier)
- If you created combined effects above, you can use those by selecting "Combo1", "Combo2", etc. You can also select the individual "MX" effects from the drop list. See the list above for an explanation of the different effects and where you'd usually want to assign them.
- Since the light strips are RGB devices, the step above will actually set ports 1 through 3 as a group, for the Red, Green, and Blue channels for the device. So the second strip will be assigned to port #4, the third will be assigned to port

#7, etc.

- Set **Port 4** to the effect that you want to assign to your second light strip (the strip connected as "Strip #2")
- Repeat for each additional light strip

Troubleshooting the "all white" problem

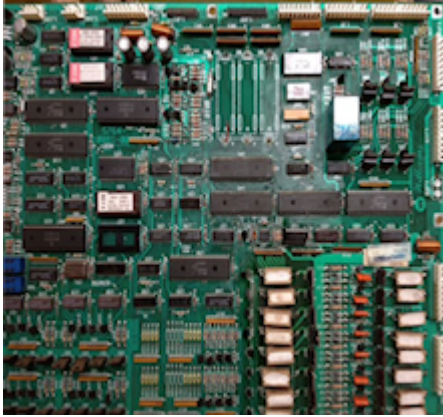
A common problem that many people seem to experience when first building an addressable light strip is that all of the lights turn on full white.

This is a result of a missing external pin connection on the Teensy. I think this varies according to the version of the Teensy you're using, so you should probably test first to see if you do indeed have the "all white" LED problem. If so, the solution is to add a jumper wire that shorts together pins 15 and 16 on the Teensy.

Here's a thread on the forums about it:

DOF addressable LEDs "all white" problem [SOLVED]

Part Four. *A Crash Course in Electronics*



66. Electronics Overview

If you're planning to build any of the add-ons for the Pinscape controller, it'll help to know a little about electronics. The next few chapters will try to get you up to speed on the basics. This isn't by any means a complete introductory course in electronics; it's more of the Cliff's Notes version. I've tried to keep the focus on the practical aspects of building the Pinscape projects.

I've tried to organize this material to be useful both as a tutorial and as a reference. If you want to bring yourself up to speed on some basic electronics before embarking on a big project like assembling the Pinscape expansion boards, hopefully the following chapters will help. On the other hand, if you want to get straight to the assembly process, you can dive right in, and come back here to look for answers if you run into any questions. To help navigate this as reference material, the Chapter 69, Field Guide to Components chapter provides a quick visual guide to many common electronic parts (including most of the parts used in the Pinscape projects), with pointers to the relevant chapters. That should help you identify any mysterious parts you have on hand and let you find out more about them, as well as help you figure out what a particular schematic symbol means.

67. Static Electricity Precautions

Some electronic parts are hyper-sensitive to static electricity. You can damage them just by touching them, if your body is charged up with static. This is especially likely in dry climates - if you live somewhere dry, you know how often you get zapped touching metal fixtures. And even if you don't notice blue sparks flying around, you can still have enough static charge on your body to damage IC chips and other especially sensitive devices.

There are some simple precautions you should take to avoid this danger when building circuit boards.

Which parts are sensitive?

An easy way to tell that a particular part is static-sensitive is to look at its packaging. If it comes in one of those silvered mylar bags, it's probably sensitive to static.



The parts that are typically most sensitive to static are:

- IC chips
- Transistors
- MOSFETs
- Diodes
- LEDs
- Circuit boards with any of the above

Precautions

First, leave parts in their silvery bags until they're needed, and keep the bags closed. The bags provide good protection, so it's best to keep the parts there when you're not actively working with them.

When it comes time to install parts on a board, you obviously have to take them out of the bags. The basic rule for handling static-sensitive devices is to **ground yourself** early and often.

The term "grounding" is pretty literal. We're not talking about some kind of meditation exercise. We're talking about literally connecting yourself electrically to the soil around your house, to dump any excess static charge on your body into the

ground. The Earth is basically an infinite reservoir of neutral charge, so excess charge naturally flows into the ground if given a path.

In practical terms, the thing to look for is an appliance with an unpainted metal case and a three-prong plug. Metal appliance cases are almost invariably wired to the third prong in the AC plug, which in turn connects in your house wiring to a big stake somewhere under the foundation that's driven into the dirt. As long as an appliance is plugged into a three-prong outlet, its metal case should be grounded.

A PC tower case is a pretty reliable example of this. The metal back plate (assuming it's unpainted) is a great grounding surface.

You don't have to keep one hand on a grounded surface the whole time you're working, although that would certainly be the ideal. A momentary touch is enough to discharge any static you've accumulated. Repeating this every few minutes while you're working will prevent new charge from building up.

Some rules of thumb:

- Ground yourself before taking a part out of its anti-static bag
- Ground yourself again any time you get up and walk around
- Ground yourself again every few minutes in any case
- Avoid wearing static-prone clothes like wool sweaters while working
- Work in an uncarpeted area

Grounding straps

You can buy a grounding wrist strap that keeps you continuously connected to ground while working - the equivalent of keeping one hand on a metal surface the whole time, but way more practical. A ground strap consists of a conductive bracelet that you wear around your wrist, and a wire that connects the strap to something grounded. Most types have an alligator clip at the other end of the cord that you're meant to attach to a grounded metal surface, so they assume that you have a grounded appliance or metal surface near your work area. You can also find grounding straps that plug into a three-prong outlet directly.

If you live in a dry climate, or you do a lot of electronics work, you might consider one of these. It's probably a bit more trouble than it's worth if you're only doing occasional hobbyist work. Maybe I've just been lucky, but in my experience, it's adequate to use the "touch a metal surface" technique as long as you're diligent about it.

68. Circuitboard assembly tips

Assembling a circuit board can be a little intimidating if you haven't done it before, especially one with as many parts as the Pinscape expansion boards. But remember that the whole point of a printed circuit board is to make it easier to build a circuit. Ideally, a PCB turns circuit assembly into a paint-by-numbers project.

Order of assembly

For most circuit boards, the Pinscape boards included, the order of installing the parts really doesn't matter. Most PCBs have simple flat layouts, without any parts layered on top of others.

The only thing I'd add is that it can sometimes be a little easier if you start with the smaller parts and save the larger ones for last, especially the taller ones that stick out the most. A couple of taller parts close together can create a little valley, where it might be harder to see the board markings for the parts that go in between.

Reading the markings on the board

The markings on the board are designed to make it easy to figure out where each part goes. Each part generally has three things marked:

- An outline showing the "footprint" of the part on the board. This is basically the way the part looks when installed if you look at it from straight overhead.

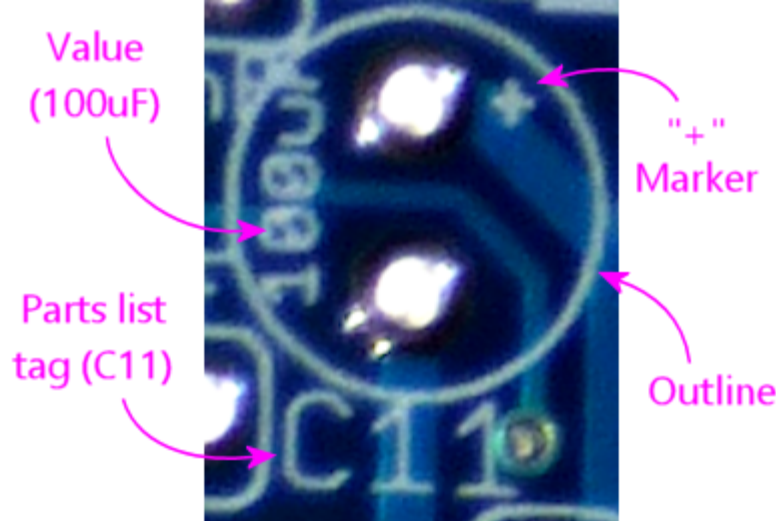
The outline often includes something to help you orient the part correctly. For example, a MOSFET's outline shows a thick bar on the side where the MOSFET's big metal heat-sink fin goes. You just have to line up the metal fin with the bar mark, and the part will be oriented properly. Similarly, a polarized capacitor's outline includes a "+" sign on the pad where the "+" leg goes.

- The part's "reference designator" (or just "designator") that identifies it in the parts list, like R1 for a resistor or C7 for a capacitor. To find the part to install, you just look up the designator in the parts list, and the parts list will tell you exactly which part to use.

("Reference designator" seems to be the formal term for this. I realized when writing this material that I didn't know what this was really called before; I didn't have a term in my mind for it, and just thought of it as a label or name. This seems to be true for most people, and indeed, even the engineer-focused EAGLE software just calls it the "name". But "label" and "name" are too ambiguous when you're trying to talk about the schematics as a language; they could be easily confused for manufacturer part numbers, or part values like "100Ω".)

- A very concise description of the part, such as the Ohms value for a resistor, the uF (micro Farads) value for a capacitor, or the part number for an IC chip. This information is redundant with the designator, since you can use the designator to look up a more complete description in the parts list, but this is there as an additional guide. It's also a good idea to cross-check the value printed on the board against what you find in the parts list, to make sure you're really looking at the right labels. Sometimes two parts are so close together on the board that you can mistake one part's labels for another's.

For example, here are the markings you might see for a capacitor (in this case, a polarized electrolytic capacitor, which has designated "+" and "-" leads):



In the chapters that follow, where we go into more detail on each of the common component types, we'll show you the specific types of circuit board markings for each type.

Installing through-hole parts

All of the parts on the Pinscape expansion boards are "through-hole" parts, meaning that they have wire leads that you insert through holes in the circuit board and then solder into place.

Here's the basic procedure to install a through-hole part:

- Identify the part's footprint on the circuit board - the little outline showing where the part goes, which should contain all of the holes for the part's wire leads
- Determine the proper orientation (see the chapters on the various component types for more on how to do this for each type)
- Straight out *or* bend the leads on the part, as needed, so that they line up with the holes in the board
- Orient the part properly and feed the leads through the holes
- Try to seat the part as close to the circuit board as you comfortably can, without forcing anything. Parts should ideally sit flat against the board, so that they can't move around or bend once installed. But it's not necessary to force anything flush with the board; sometimes the leads won't quite bend enough and the part will sit a few millimeters above the board. That's fine.
- Holding the part in place (and keeping it from slipping back further away from the board), flip the board over and solder each of the leads to the pads on the bottom of the board
- After the solder cools, use wire snippers to clip off the excess length from the leads, so that there's no dangling wire past the solder joint

Soldering tips

- See the recommendations on soldering tools in Chapter 87, Tools, including the recommendations on the solder itself. I've heard so many people on the forums say "I'm not a very good solderer", but what they often really mean (without knowing it) is "I'm using a crappy Home Depot soldering iron and crappy Home Depot solder". Those soldering irons are for casual home repair jobs, not

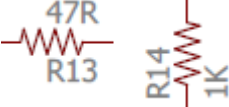



delicate electronics work. Using them for circuit boards can be painfully difficult. You'll be amazed at how much your skills improve when you have the right tools.














- Keep a wetted solder-tip cleaning sponge handy while working. A soldering station should include this. Saturate it, then squeeze out the excess water. It should be thoroughly moistened but not dripping wet.
- Wipe the soldering tip on the sponge from time to time while working to keep it clean.
- Make sure the iron is at full temperature before you begin. A proper soldering station has a readout showing the temperature and a thermostat that holds the temperature steady.
- The default temperature setting on my Hakko soldering station is 750° F, and that seems to work well for PCB soldering work.
- When the iron is hot, coat it with a little bit of solder, just enough to flow over the surface. Wipe off any excess bead on the solder cleaning sponge.
- Most newbie solderers want to use the iron to melt the solder, and then try to drag the solder onto the parts. That's the wrong way to do it.
- The *right* way to do it is to heat the *parts*.
- Start by putting the two parts to be joined into the desired position.
- With the parts held together in this position, firmly press the tip of the iron against the point where the parts meet. Hold it there for several seconds to let the parts heat up. Then touch the solder to the junction point in the parts - *not* to the soldering iron, but to the join point. If the parts are hot enough, the solder will melt and flow over the parts, forming a bubble around the junction point.
- Don't use too much solder - just enough to flow over the parts and form a bubble.
- As soon as the solder flows over the junction, withdraw the soldering tip, but keep holding the parts together, keeping everything as still as possible, for about 5 to 10 seconds while the solder cools.
- Visually inspect the solder joint to make sure that the solder is confined to the desired area, and that it completely surrounds and covers the junction point. On a circuit board, it should stay within its solder pad, and it should form a nice little droplet shape covering the solder pad and wire lead. Make sure there are no gaps and that the wire lead is covered all the way around. Wiggle the part gently to make sure that the connection to the pad is thoroughly immobilized.

69. Field Guide to Components

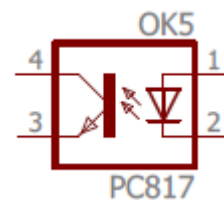
Or, how to recognize electronic parts from quite a long way away. Here's a quick visual reference guide to the major components used in the Pinscape expansion boards. We'll cover each category in greater detail in the chapters that follow.

Note that the example photos are just that - examples. All of these parts come in an almost infinite variety of shapes, sizes, and colors. These photos should at least look similar to most of the parts used in the Pinscape projects, but there's a lot of variation even in that limited scope, so don't worry if you don't see an exact picture of one of the parts you're using.

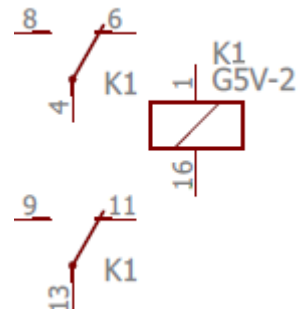
Description	Examples	Schematic symbols
Chapter 72, Resistor Resistance value in Ohms (Ω)		
Chapter 73, Capacitor (ceramic disc) Capacitance value in Farads (F); ceramic capacitors are unpolarized		
Chapter 73, Capacitor (electrolytic) Capacitance value in Farads (F); electrolytic capacitors are polarized (one lead is "+")		
Chapter 74, Diode		

Chapter 75, LED (light-emitting diode)																																																																		
Chapter 77, Transistor (bipolar)		<div><div>NPN</div><div>T8 2N4401 BC337</div></div> <div><div>PNP</div><div>T5 2N4403</div></div> <div><div>NPN</div><div>T4 2N4401 BC337</div></div> <div><div>PNP</div><div>T3 2N4403</div></div>																																																																
Chapter 77, Transistor (Darlington)		<div><div>T2 TIP122</div></div>																																																																
Chapter 78, Transistor (MOSFET)		<div><div>Q1 BUK7575-55A</div>N-channel</div> <div><div>Q2 IPP121</div>P-channel</div>																																																																
Chapter 79, Integrated Circuit (IC) chip		<div><div>TCL5940</div><table><tr><td>28</td><td>OUT0</td><td>VPROG</td><td>27</td></tr><tr><td>1</td><td>OUT1</td><td>SIN</td><td>26</td></tr><tr><td>2</td><td>OUT2</td><td>SCLK</td><td>25</td></tr><tr><td>3</td><td>OUT3</td><td>XLAT</td><td>24</td></tr><tr><td>4</td><td>OUT4</td><td>BLANK</td><td>23</td></tr><tr><td>5</td><td>OUT5</td><td>GND</td><td>22</td></tr><tr><td>6</td><td>OUT6</td><td></td><td></td></tr><tr><td>7</td><td>OUT7</td><td>+</td><td></td></tr><tr><td>8</td><td>OUT8</td><td></td><td></td></tr><tr><td>9</td><td>OUT9</td><td></td><td></td></tr><tr><td>10</td><td>OUT10</td><td>VCC</td><td>21</td></tr><tr><td>11</td><td>OUT11</td><td>IREF</td><td>20</td></tr><tr><td>12</td><td>OUT12</td><td>DCPROG</td><td>19</td></tr><tr><td>13</td><td>OUT13</td><td>GSCLK</td><td>18</td></tr><tr><td>14</td><td>OUT14</td><td>SOUT</td><td>17</td></tr><tr><td>15</td><td>OUT15</td><td>XERR</td><td>16</td></tr></table><div>IC1</div></div>	28	OUT0	VPROG	27	1	OUT1	SIN	26	2	OUT2	SCLK	25	3	OUT3	XLAT	24	4	OUT4	BLANK	23	5	OUT5	GND	22	6	OUT6			7	OUT7	+		8	OUT8			9	OUT9			10	OUT10	VCC	21	11	OUT11	IREF	20	12	OUT12	DCPROG	19	13	OUT13	GSCLK	18	14	OUT14	SOUT	17	15	OUT15	XERR	16
28	OUT0	VPROG	27																																																															
1	OUT1	SIN	26																																																															
2	OUT2	SCLK	25																																																															
3	OUT3	XLAT	24																																																															
4	OUT4	BLANK	23																																																															
5	OUT5	GND	22																																																															
6	OUT6																																																																	
7	OUT7	+																																																																
8	OUT8																																																																	
9	OUT9																																																																	
10	OUT10	VCC	21																																																															
11	OUT11	IREF	20																																																															
12	OUT12	DCPROG	19																																																															
13	OUT13	GSCLK	18																																																															
14	OUT14	SOUT	17																																																															
15	OUT15	XERR	16																																																															

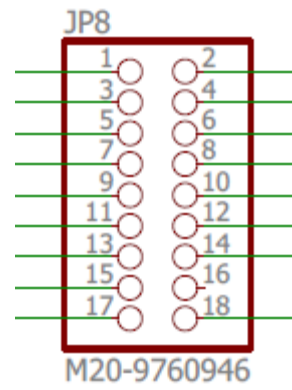
Chapter 79,
Optocoupler



Chapter 76, Relay



Chapter 81, Pin
headers

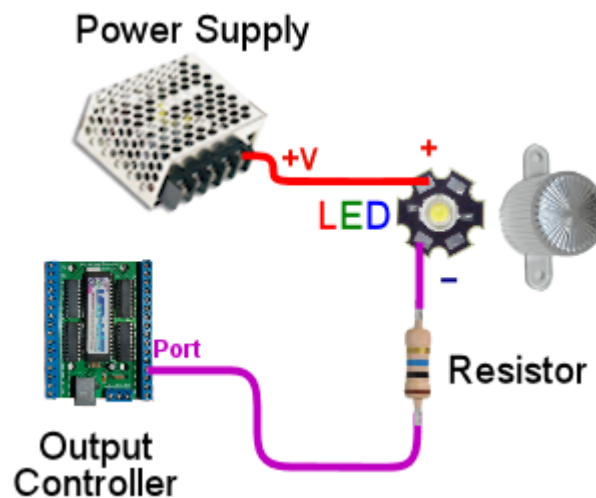


70. Schematics

A schematic is a map of a circuit showing the components and how they're connected to one another.

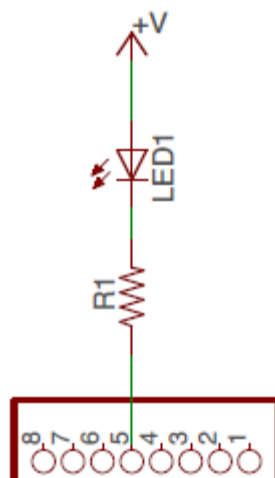
You don't have to learn how to read a schematic to build the Pinscape Controller projects, but it's a skill that might come in handy if you need to troubleshoot one of the Pinscape boards after building it. So this section provides a very quick introduction, hopefully just enough that you can make sense of the Pinscape schematics should you ever need to look at one.

Schematics use their own special symbolic language, the way that music has its language of staves and notes, so they can look pretty opaque at first. A lot of electronics-for-newbies tutorials try to avoid the formalism of schematics by using pictorial circuit diagrams. I've even used a few of those in this guide, like this one from the Chapter 52, LED Resistors chapter, showing how to wire a current-limiting resistor into a flasher LED circuit:



That's fine for simple circuits, but it doesn't "scale up" well to large circuits with many components. It's also a bit fuzzy, in that the little pictures of the parts could be mistaken for something else, especially if there were more than a few distinct parts in the diagram. Warm and fuzzy might be fine for artists and puppies, but engineers don't like fuzzy. They like clarity and precision.

Schematics were invented as a more concise and precise way of showing this kind of information. To a large extent, a schematic is a lot like the pictographic representation above; the big difference is that we replace the pictures of the parts with symbols representing the parts. Here's a schematic version of the circuit above:



The symbols give schematics their precision, since there's a standard set of them that everyone agrees on. They constitute a sort of vocabulary, so you'll find yourself able to read most schematics pretty readily once you learn the basic symbol set.

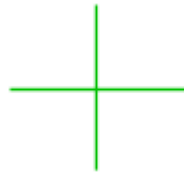
Wires

The simplest schematic symbol is a wire, which is shown as a line between two parts.



This schematic idea of a "wire" is an abstraction - it doesn't necessarily represent a literal piece of wire. And you'll notice it doesn't say anything about how long the wire is or where it goes on the circuit board. A schematic "wire" just represents *some* type of electrical connection. In the physical realization of the circuit, the schematic "wire" could be an actual piece of wire, or it could be a copper trace on a printed circuit board. The schematic wire is just saying that the two points are connected electrically, with the details left up to whoever builds the circuit physically. (Which also means that you could build a functionally identical circuit in different physical arrangements: on a printed circuit board, on a breadboard, with a bunch of loose parts and wires...)

When wires run directly into components as shown above, it means that the components are connected to the wires. When wires cross over each other, though, they're *not* automatically connected. Here are two separate pieces of wire that aren't electrically connected to each other:



Two separate wires, not connected to each other

Some schematics show non-connected wire crossings with a little hump:

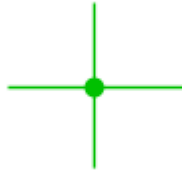


Another way of showing a non-connected wire crossing

The little hump is to make it more explicit that there's no connection. But this seems to be more of a "beginner" convention that you don't see much in engineering schematics. The modern practice is to use simple straight crossings.

Whenever the wires at a crossing *are* connected, we add a big dot to indicate the

connection:



Two wires connected together

The dot is also used at any connected "T" junction:



Cross-references

When a schematic reaches a certain level of complexity, you get so many wire lines going across such large swaths of the diagram that it gets hard to follow them all. So there's a convention that greatly reduces the tangle of wire lines by removing long lines and replacing them with "cross-references".

A cross reference point looks like this:

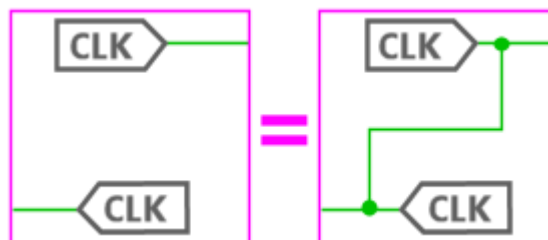


The word inside the arrow-shaped box is the cross-reference label. Sometimes you see it with the arrow box, and sometimes you see it with just the text label:



They both mean the same thing. I like the box notation because it makes it easier to spot these points at a glance, but some engineers don't bother with them. It's like the little humps to represent explicit "not connected" crossings: a text label by itself at the end of a wire can only mean one thing, so some engineers see the little box as redundant.

In either case, what this means is that this point in the wiring is connected to *all of the other points in the wiring* that have the same label. This lets you connect two points on opposite sides of the schematic without having to route a green line all the way across the page.

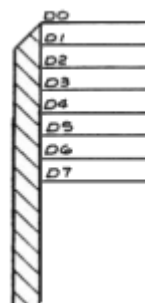


Now, that's just an oversimplified example - when the two "CLK" points are close together like that, you'd usually just draw the wire. But imagine if those two "CLK"

The label in a cross-reference is just an arbitrary name defined by the person who drew the schematic. It doesn't mean anything within the schematic language the way that a resistor symbol means something; it's just a name for that connection point, like the name of a street. These names are usually chosen to be somehow descriptive, but that's purely within the context of the particular circuit.

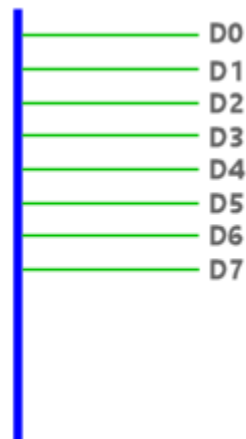
Buses

On the old Williams pinball schematics, buses are represented like this:



The EAGLE notation for a bus is a little different. I personally prefer the old Williams

notation, since I think it's clearer, but the EAGLE approach is really the same idea once you learn what it looks like. EAGLE's way of drawing this is to just use a thick blue line to represent the bus.



Some important things to note about the bus notation:

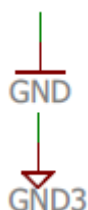
- Unlike regular wire-to-wire connections, there are no "dots" to indicate the connection points. Any regular wire that ends at a bus line is connected to the bus.
- The wires going into the bus **aren't** connected to each other. D0 through D7 are still all separate wires. This is just short-hand to show the whole group of wires as a single line/bar rather than having to draw all of them individually.
- If you want a physical analogy, you can think of the bus as a shrink-wrap tube that wraps around all of the wires making up the bus.

Ground connections

"Ground" has several meanings in electronics, so you see different symbols for it. The most common symbol you see is this, which typically represents an "Earth" ground, meaning literally a connection to the soil, usually through the ground prong in your house's power wiring.



In the EAGLE schematics, we don't use that exact symbol, and we don't have any points where we're talking about the literal Earth ground. You'll see these two symbols in the EAGLE schematics instead:



In our schematics, these are what's known as DC grounds. If you think about a power supply as though it were a battery, it would have a "+" post and a "-" post. In that way of thinking, the "-" post corresponds to the DC ground. That's not quite the

way engineers think of power supplies, though: they think of what you'd call the "-" end of the battery as 0V for "zero volts". That's the reference point, and all of the other supply voltages are relative to that reference point - so the disk connectors on an ATX power supply, for example, have a +5V supply line and a +12V supply line, relative to that 0V. This 0V point is what we call "ground" in a DC circuit.

Why do we have two different ground symbols, and what's "GND3"? I'm sure you already guessed that "GND" is an abbreviation for "Ground". "GND3" stands for "Ground 3", which is a separate DC ground point in the circuit from the regular "GND". You'll see "GND2" in other places, which is a second one.

The Pinscape schematics use the multiple grounds for two reasons.

- The first is what you might expect, which is to isolate different parts of the circuit. The expansion boards are set up to isolate the "logic" part from the "power" part, by using separate power supplies for the two sections. The regular "GND" point is the 0V ground connection for the PC power supply (the "logic" section), and "GND1" is the corresponding connection for the secondary power supply that powers your knocker coil and shaker motor (the "power" section).

We use the two separate symbols to suggest this separation visually.

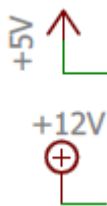
- The second is an inelegant way handling some special needs of the circuit boards. GND1, GND2, and GND3 in these schematics are actually all connected together. They're given separate names because that lets us persuade EAGLE to given them different trace widths on the circuit boards, mostly so that some of the connections can handle high current loads.

All of the connected ground points use the same "triangle" symbol, which hopefully helps suggest the connection visually.

In all of these cases, the GND points are ultimately connected to the Ground connection on a power supply unit. For a PC-style ATX power supply, the "ground" connection is the black wire in all of the disk cables coming out of the unit.

Power connections

As with the grounds, the expansion boards use two symbols to represent power supply connections:



We use the two symbols for the same reason that we do with the grounds: because the expansion boards are designed to be connected to two separate power supplies. One symbol, the little arrow, represents the main PC power supply. We use the circle-plus symbol for the secondary power supply.

The power supply connections are labeled with the voltage.

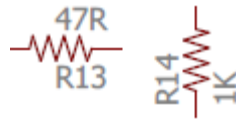
Just to be clear, these power supply symbols represent power **inputs**, where you connect the circuit boards to a separate power supply unit that supplies the labeled voltage. (As opposed to representing power outputs where the boards are generating

power for something else. That's not something we do in any of the Pinscape boards.)

Resistors

A resistor is a simple component that adds electrical resistance (analogous to friction in a mechanical system) to a circuit. See Chapter 72, Resistors.

The symbol on a schematic is a jagged line.



The version on the right is the same as the version on the left, just rotated 90°. We wanted to show that just to clarify that it means the same thing no matter how it's rotated. The same is always true for all other component types. Schematic writers will orient each symbol as they see fit for legibility.

A resistor on a schematic is usually accompanied by two labels, usually placed on either side of the resistor symbol.

The first is an "R" followed by a number - in the example above, R13 and R14. This is formally called the "reference designator" for the resistor, or just the "designator". It's an arbitrary, unique identifier for the part, primarily for cross-referencing to the parts list. It has no meaning by itself; it's just a name. The "*Rnumber*" notation is just a convention, too; in principle any sort of label would do. But the "R" labeling for resistors is practically always used. Designators always have to be unique throughout the schematic, so that you can identify each individual physical part.

The second label is the resistance value in Ohms. This is usually written in one of these formats:

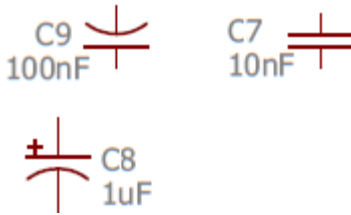
- **47R** means **47 Ohms** - the "R" suffix is usually used instead of the real symbol for Ohms, Ω , probably because the Ω symbol could be mistaken for a zero, or maybe just because it was hard to enter the Ω symbol in older software
- **4R7** means **4.7 Ohms** - an "R" sandwiched between numbers like this stands in for a decimal point; this notation is used because real decimal points aren't always legible in crowded areas or tiny fonts
- **47K** means **47 Kilo Ohms** = 47 k Ω = 47000 Ohms; the "K" means "times a thousand Ohms"
- **4K7** means **4.7 Kilo Ohms** = 4.7 k Ω = 4700 Ohms; as with the embedded "R", an embedded "K" replaces a decimal point, and *also* still means "times a thousand Ohms"
- **47M** means **47 Mega Ohms** = 47 M Ω = 47,000,000 Ohms; "M" means "times a million"
- **4M7** means, you guessed it, **4.7 Mega Ohms** = 4.7 M Ω = 4,700,000 Ohm

A resistor has two connections to the outside world. The schematic symbol shows this as a straight line sticking out of each end. Resistors aren't polarized, meaning the two ends are interchangeable. There's nothing in the symbol indicating which way the resistor goes because it doesn't matter which way it goes.

Capacitors

A capacitor is a simple component that adds electrical capacitance to a circuit, which is similar to a (very) tiny rechargeable battery. See Chapter 73, Capacitors.

The symbol for a capacitor consists of two parallel lines separated by a small gap, or one straight line and one curved line next to each other. In some cases, there might be a little "+" sign adjacent to the straight line.



As with resistors, each capacitor in a schematic is typically accompanied by two labels.

The first label is a "C" followed by a number. This is the capacitor's reference designator - an arbitrary ID for the part, purely for looking it up in the parts list. It's the capacitor equivalent of the "R" number for a resistor. It doesn't have any meaning by itself; it's just a name to look up in the parts list. Reference designators always have to be unique throughout the whole schematic, so that you can uniquely identify every physical part that goes into the circuit. Note that there's no absolute rule that a capacitor's designator has to start with "C", but almost everyone uses that convention, so it's practically a rule.

The second label is the capacitance value in Farads. This is almost always in one of the following formats:

- **100pF** means **100 pico Farads** or 100 trillionths of a Farad
- **100nF** means **100 nano Farads** or 100 billionths of a Farad
- **100uF** means **100 micro Farads** or 100 millionths of a Farad (this is more properly written **100μF**, but the Roman alphabet "u" is usually used instead because of pervasive ASCII chauvinism in computer software)
- **100mF** means **100 milli Farads** or 100 thousandths of a Farad (these are extremely large capacitors that you rarely see in micro-electronics, but you might see one in a power supply; there's a 30mF capacitor in my *Whirlwind*'s lamp power supply circuit, and it's about the size of a soda can)

A capacitor has two connections, represented in the symbol by the lines coming out of either end.

If there's a "+" sign in the symbol, the capacitor is a "polarized" type, meaning that one end has to be connected to the positive voltage and the other end is for the negative voltage. The "+" sign in the symbol marks the end that connects to the positive voltage.

If there's no "+" sign in the symbol, the capacitor is an "unpolarized" type, meaning it doesn't matter which end connects to which voltage. The two ends are interchangeable (like in a resistor).

The polarized or unpolarized status is a function of the physical type of capacitor you're using. If the schematic symbol has the "+" sign, you **must** use a polarized capacitor in the physical build. If not, you **must** use an unpolarized capacitor. You can generally tell if a particular physical capacitor is polarized by looking at its

material type:

- A **ceramic disc capacitor** is always unpolarized
- An **electrolytic capacitor** is always polarized

There are several other types besides these, but these are the only types you'll see in the Pinscape boards. Most of the other, more exotic types are non-polarized, including film and glass capacitors. Tantalum capacitors are a type of electrolytic capacitor, so they *are* polarized.

Diodes

A diode is a semiconductor that only lets current flow in one direction, sort of an electronic one-way valve. See Chapter 74, Diodes.

The symbol for a diode on a schematic is an arrow with a bar:



Each diode on a schematic is typically accompanied by two labels. The first is a "Dnumber" label giving the reference designator, for looking up in the parts list. As with resistor "R" numbers and capacitor "C" numbers, this has no meaning by itself; it's just an arbitrary ID for cross-referencing with the parts list. Almost everyone uses "D" for "diode" in these labels by convention.

The other label is the *semiconductor identifier* for the type of diode to be used. This is sort of like a manufacturer part number or catalog number, but it's not specific to any one manufacturer; it's a generic descriptor system that the industry uses. Diodes don't have a simple "unit" that describes them like Ohms for resistors or Farads for capacitors, so schematic writers use this semiconductor ID to specify which part they want you to use. For a diode, this usually starts with "1N", as in the example above, **1N4007**. You can use this ID on sites like Mouser to search for matching parts to buy.

Diodes are inherently polarized, so they have to be wired into the circuit in the correct direction. If you put a diode in backwards, it won't work properly (and might do damage). The direction is indicated by which way the arrow is pointing. On the physical diode, you should see a stripe painted on one end; that stripe corresponds to the bar that the arrow is pointing to in the schematic symbol.

LEDs

An LED is actually just a special case of diode. That's the "D" in the acronym - "Light Emitting Diode" - and it's quite literal. The schematic symbol for an LED is therefore basically the same as the symbol for a regular diode, with an embellishment to indicate that it's the special light-up kind: a couple of little arrows representing the photons flying away.



An "LEDnumber" reference designator usually takes the place of the "D" designator for a regular diode, but there's less of a universal convention about this, so you

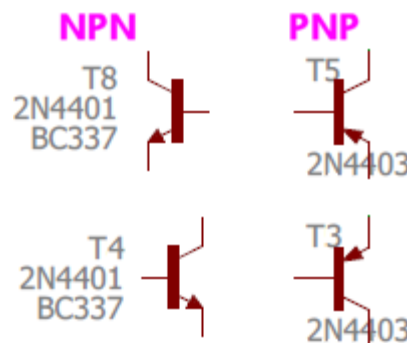
might see other formats. You should always see some designator, though, for looking up in the parts list.

And as with a regular diode, an LED symbol will often be accompanied by some sort of formal part ID, such as a manufacturer part number, to tell you what to buy. This might not be present in the schematic, though, in which case you'll have to check the parts list.

Transistors - bipolar

A bipolar transistor (or bipolar junction transistor, BJT) is a common type of transistor that's used in all sorts of circuitry as a small amplifier or an electronic switch. See Chapter 77, Transistors.

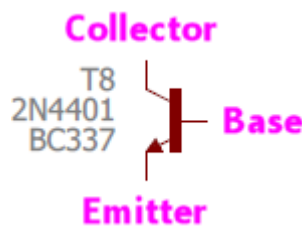
The symbol for a transistor consists of a thick bar with three lines sticking out, one straight line on one side, and two diagonal lines on the other side. One of the diagonal lines has an arrow, which might point towards or away from the middle bar.



If the little arrow points *away from* the bar, the symbol represents an "NPN" transistor. If the arrow point *towards* the bar, it's a "PNP" transistor.

Note that the little arrow might be shown at top or bottom, and it might be on the left side or the right side. None of that makes any difference - the symbol means the same thing no matter how it's flipper or rotated. Schematic writers will flip the symbol top-to-bottom, or left-to-right, or rotate it at different angles, according to what's convenient to make the lines between nearby connections shorter. It doesn't change the meaning.

The three lines represent the three connections to the transistor, called the base, collector, and emitter:



- The straight line by itself on one side is always the **base** or **B**
- The diagonal line with the arrow is always the **emitter** or **E**
- The other diagonal line is always the **collector** or **C**

On some schematics, the whole thing will be enclosed in a circle:



The circle doesn't change anything; it's just an alternative way of drawing the symbol.

Transistors have parts list tags just like other components. These most commonly start with "T" or "Q". As with the "R" tags for resistors and "C" tags for capacitors, these are just arbitrary tags to look up in the parts list, with no other meaning.

Transistors are also usually labeled with the semiconductor ID, like a diode is. In the case of a transistor, this usually starts with "2N". You might also see other part numbers, such as the "BC337" in the examples above. When two numbers are listed for one part like this, it indicates *alternative* parts that you can use - so in the case of T8 above, you could use a 2N4401 or BC337 interchangeably.

Transistors have to be inserted into the circuit with the three prongs in exactly the right order. As with diodes, each prong has a different function, and the part won't work if it's not inserted correctly. There's no standard way of marking a physical transistor to indicate which leg is which - the only way to tell is to look it up in the manufacturer's data sheet. In the case of the Pinscape expansion boards, though, you can tell how to orient the part from the looking at the silk-screened markings on the circuit board; we'll explain that in Chapter 77, Transistors.

Transistors - Darlington

A Darlington transistor is a variation on the basic bipolar transistor that combines two bipolar transistors in one physical package, for greater amplification and power handling than a regular bipolar transistor can handle. See "Darlington" in Chapter 77, Transistors.

For the purposes of building the Pinscape boards, Darlington's are the same in every respect as bipolars. But they have a different symbol in a schematic, so we wanted to show you what that looks like so that you can recognize it when you see it:



The symbol is pretty literal - it looks like two regular transistors connected together, because that's just what a Darlington is. A Darlington still has the same three external connections (base, collector, and emitter).

Transistors - MOSFET

A MOSFET is another kind of transistor constructed in a different way from a bipolar transistor. It performs the same transistor functions as a bipolar, but the electrical characteristics are somewhat different, so it has its own representation on a schematic:

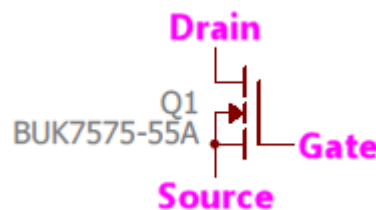




As with bipolars, there are two types of MOSFETs, known as N-channel and P-channel MOSFETs. The schematic symbols for the two types are almost the same, befitting their similar construction and behavior, with one subtle difference: the direction the arrow points in the middle of the diagram. In an N-channel MOSFET, the arrow points inwards, into the middle section; in a P-channel MOSFET, it points outwards.

MOSFET symbols in a schematic are labeled like other transistors, with a reference designator (we're using a **Q** prefix here, but you might also see a **T** prefix) and a part number. For MOSFETs, this is almost always a manufacturer part number, so there won't be any particular pattern to it; it'll just be an alphanumeric string that you can look up on Mouser and in other vendor catalogs.

Like bipolar transistors, MOSFETs have three prongs with distinct functions, and they have to be oriented properly when installed. The prongs of a MOSFET go by different names from a bipolar's legs:



- The **Gate** is the prong off by itself on one side
- The **Source** is the prong that connects to the arrow
- The **Drain** is the remaining prong

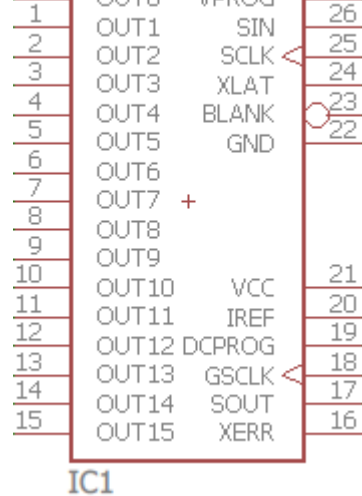
As with bipolar transistors, there's no standard marking system to identify which leg is which on the physical part; you just have to look it up in the manufacturer's data sheet. The Pinscape expansion boards show how the part is oriented on the silk-screened markings on the circuit boards.

IC chips

Integrated Circuit (IC) chips are complex devices consisting of many components packed into a single package. See Chapter 79, IC Chips.

ICs are extremely diverse in function and physical packaging, so it's not entirely fair to lump them all into a single category. But there are enough commonalities to how they're handled in schematics that we can make some practical generalizations. For our purposes, an IC is a bit of circuitry all packed into a discrete physical package, with multiple connection points ("pins" or "leads" coming out of the physical chip). The schematic treats an IC as a "black box": a bunch of wires connect it to the outside world, but what's inside is of no concern in the schematic. As a result, the schematic symbols for ICs look pretty much like empty boxes:



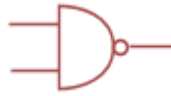


Here are some features to note:

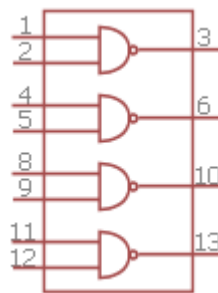
- The overall IC package is represented by a rectangular box
- We're using the term "black box" figuratively, as you can see that we haven't literally drawn the box in black ink here; "black box" is a metaphor that engineers use to talk about something with complex inner workings that we don't have to see (or understand) in order to use it
- The wires coming out of the box represent the "pins" or "leads" on the physical IC, which are the electrical connection points
- Different ICs have different numbers of pins, so you might see boxes like this with three wires coming out (such as for a voltage regulator), or a couple hundred wires (for a CPU chip), or anything in between
- The positions of the wires around the perimeter of the box don't correspond to the physical layout; this is just an abstract representation, like any other schematic symbol
- The order of the wires in the symbol doesn't reflect the ordering of the pins on the physical chip - for that you need to consult the the little number written adjacent to each wire just outside the box, which tells you the pin number on the physical chip that this wire corresponds to
- The labels written on the *inside* of the box adjacent to the pins are mnemonics for the functions of the pins; these are purely for convenience, to help you remember the function of each pin without having to keep going back to the chip's data sheet
- The schematic symbol will usually be accompanied by a reference designator, analogous to "R5" for a resistor or "C7" for a diode; for an IC, it's usually of the form "IC10", but lots of other prefixes are used, including odd ones like "U\$" - engineers started running out of unique letters for these tags at some point, so they resorted to other symbols. The prefix might also be specific to the type of IC; for example, the Pinscape schematics use "OK" for optocouplers.
- The schematic symbol will also usually be accompanied by the manufacturer part number for the specific IC ("TLC5940" in the case of this example); some of these are generic part descriptors for parts made by many manufacturers, while others are manufacturer-specific

There are exceptions to this "black box" treatment. Some types of ICs have specific functions that are so commonly used that they have their own unique schematic symbol that's more representative of the function. We'll see this for optocouplers below. Other examples include common logic gates, such as NAND and NOR gates and inverters, which sometimes (but not always) are shown with special logic

symbols in place of the plain box. The Pinscape boards don't use any of those types of symbols, but you might see them in other schematics. For example, a NAND gate might be drawn like this:



That's a very specific notation that engineers recognize as a NAND gate, so it's sometimes used in place of the more generic "black box" notation for miscellaneous ICs. But you might just as well see the plain black box notation; it's really up to the schematic writer. You might even see a hybrid notation that shows the NAND IC as a black box, and then *also* draws the logic symbol inside the box. This is just a more pictographic equivalent of the little mnemonic labels that you see in the TLC5940 diagram above, since it shows you the function of each pin visually. For example, here's how you might see a chip that contains four NAND gates represented:

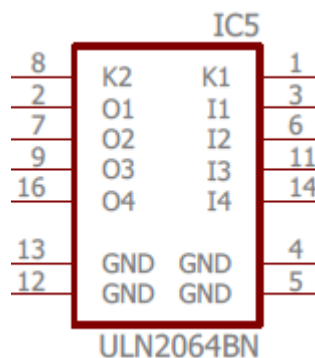


Multi-gang chips

There's another "advanced" convention that you should know about when it comes to IC chips, and even some other types of components (such as relays, as we'll see below).

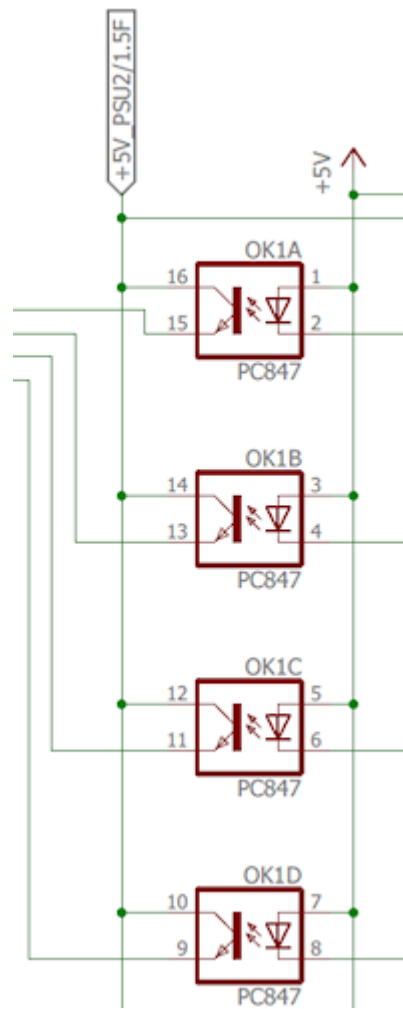
Some chips come with two or four or eight copies of the same basic building block. For example, the Pinscape boards use the PC847 chip, which consists of four separate optocouplers on one chip. That makes it a "quad optocoupler" chip. Pinscape also uses a "quad Darlington" chip, the ULN2064, which consists of four Darlington transistors on one chip. This is quite common with basic components like logic gates, optocouplers, op-amps, and transistors.

In schematics, one way to represent these multi-gang chips is the generic way we saw above, where you draw a big box for the entire chip. The ULN2064 uses a generic symbol like that in the Pinscape schematics:



So nothing new so far! But now we come to the novel part. Sometimes, rather than using the generic black box format, schematics will represent a multi-gang part with

its individual building blocks all separated from one another, *as though they were separate components*. EAGLE uses this approach for the PC847, that "quad optocoupler" we mentioned. Rather than drawing it as a big box with 16 pins coming out of it - which is, in fact, exactly what the *physical* package looks like - EAGLE draws this as though it were four separate optocouplers. Here's an excerpt from the Pinscape "main interface board" showing one PC847 broken up into four optocoupler symbols:



If you didn't already see what we meant about how schematic symbols are "abstract", this probably really drives it home.

Two questions: Why in the world do they do this? And how are you supposed to tell that those four boxes are really one physical IC chip?

First the "why". They do this to make the schematics more readable. I know, it can seem like it does the opposite. But if you think about it in terms of understanding what the circuit *does* as opposed to how to build it out of parts, this representation is actually a lot more useful than drawing all of those 16 wires going into a black box. With this format, you can plainly see which wires control the LEDs and which wires are connected to the photo-transistors.

There's another benefit that's not even apparent in this picture, too. Those four boxes representing the individual optos don't have to be grouped together in one place in the schematic - they can be split up and spread out. They really are four separate boxes as far as the schematic is concerned. This allows the schematic writer to place each one close to the other components it's connected to, so that the wire connections are shorter and easier to follow. The Pinscape boards keep all of the groups like this together, but if you look at some old Williams pinball schematics,

you'll find that they take ruthless advantage of this ability to spread the parts around. You'll find quad NAND gate chips with the individual gates on different pages, and dual op-amp chips with the individual op-amp blocks likewise widely separated.

Now to the second question, how you're supposed to relate the four boxes back to one physical chip. The trick is to look for matching tags. You can see that each individual opto in the diagram above has its own separate set of labels - each one is tagged "PC847" (the chip name) and "OK1x" (the reference designator for the parts list lookup) - as though it were a standalone part. The designator is what gives away the secret that all of these "OK1" elements are part of the same physical chip. And why is that? Because a designator is **always** unique across the whole schematic - like Highlander, there can be only one OK1. The fact that the same designator appears on four symbols means that the symbols are all portions of the same physical component.

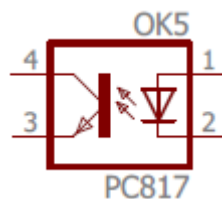
Okay, back up a sec... I'm sure you noticed that these *aren't* actually all labeled "OK1". They're labeled OK1A, OK1B, OK1C, and OK1D. But when have we ever seen a letter *after* a number in a designator before? Never. Tags until now always ended in a number. You've probably already guessed what it means when you add a letter to the end: it means that we're talking about a sub-block of a multi-gang chip like this. The physical chip is still called "OK1", but they've given these additional "A" through "D" suffixes to the individual optos within OK1 to help us distinguish them.

Those A-B-C-D suffixes aren't always used, by the way. They're used for this particular opto, but for other types of chips, you might just see the same base designator repeated on each block. Each block might be tagged, say, IC9, with no suffix. The A-B-C-D suffixes aren't really all that necessary, since you can still tell which block is which in physical terms by looking at the pin numbers. If you look at OK1A through OK1D above, you'll see that each pin is still numbered in terms of the overall 16 pins of the physical chip, the same as if it were the black-box kind of symbol.

Optocouplers

An optocoupler is a special kind of IC chip that connects two parts of a circuit via a photo-emitter and a photo-receiver. This provides electrical isolation between the two parts of the circuit while allowing them to transmit information across the interface. See "Optocouplers" in Chapter 79, IC Chips.

As we mentioned above, some types of IC chips are so commonly used that they get their own schematic symbols. The optocoupler is one of these special cases. An optocoupler looks like this on our schematics:



That's a little like the NAND gate example we talked about above, in that it starts with the generic IC "black box", but then adds a pictograph inside to depict what the pins do. An optocoupler internally consists of an LED (usually infrared) and a phototransistor (a special type of transistor that's activated when light hits it, rather than being controlled by an electrical signal), so you see the symbols for an LED and a transistor. Just as with the mnemonic text labels inside the black box on our

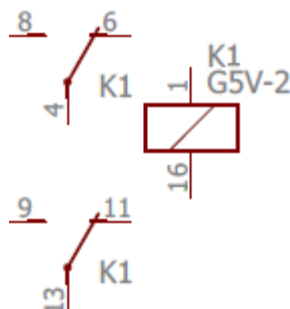
TLC5940 example earlier, the pictogram is just a mnemonic to help you remember what the pins do. In terms of actually using the chip physically, you can really ignore all of that, since all you have to pay attention to is the pin numbers written on the outside wires - exactly like any generic IC chip.

Note one other deviation from more generic ICs: the Pinscape schematics use "OK" as the designator prefix instead of "IC", so in this case, "OK5" instead of "IC5". You might also see "OC" prefixes in other people's schematics.

Relays

A relay is an electrically-controlled mechanical switch. An electromagnet in the relay operates a little rocker switch that connects and disconnects another circuit. See Chapter 76, Relays.

The schematic symbols for a relay vary. Here's the format that we use in the EAGLE plans for the Pinscape boards:



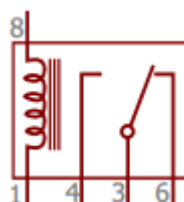
This is a little hard to parse, because it breaks up the relay into its component parts:

- the little box at the right represents the electromagnet
- the two clusters at the left represent the mechanical switches (this particular relay has two of them, because it's a "double pole" relay, meaning it has two electrically independent switches operated by the same electromagnet)

This follows exactly the same convention that we saw for some IC chips with multiple repeated blocks - see "Multi-gang chips" above for more on that. The short summary is that we can tell that these three little blocks are actually part of the same physical relay from their tags. They're all tagged "K1". Since a designator must be unique across the whole schematic, the fact that we have three things tagged K1 can only mean that they're all part of the same physical component.

As with an IC chip, the numbers on the connection points indicate the pin numbers on the physical relay. There aren't really any conventions for how the pins on a relay are numbered; it's just something you have to look up in the data sheet for the individual device.

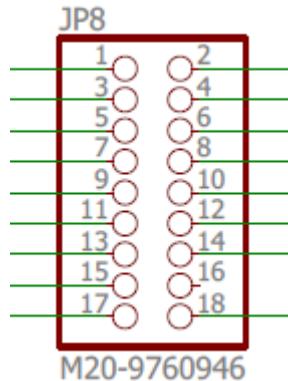
You might also see schematic symbols for relays that are more literal, with a pictograph for the electromagnet coil, and the whole thing enclosed in a black box like an IC chip. For example:



Connectors and pin headers

Circuit boards need connections to the outside world, usually in the form of some kind of plug-in connector. We provide an overview of some of the most common types, and the ones we use on the Pinscape boards, in Chapter 80, Connectors.

On a schematic, we draw connectors like this:

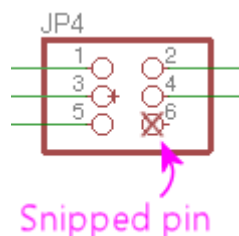


It looks a bit like a generic IC, but note that the wires all go into the box and connect to little circles. The circles represent the pins on the connector. The numbers next to the circles are the pin numbers, which tell you which pins they correspond to on the physical connector. The pin numbering conventions are different for different parts; we explain our conventions in Chapter 80, Connectors. As always with schematic symbols, the order and arrangement of pins shown in the schematic doesn't necessarily correspond to the physical pin layout, so you have to pay attention to the pin numbers.

On the Pinscape schematics, most connectors have a "JP" label, for "jumper", as in JP7 or JP15. (Another common convention that you'll see on other schematics is a simple "J" prefix, such as J9.) As usual, this is the reference designator, for looking up the connector in the parts list, and (as usual) it has no meaning other than to serve as a cross-reference. You might also see a manufacturer part number, as in the example above. Some of the connector types are generic enough that you can substitute equivalent parts from other manufacturers, so the part number might only be a suggestion to help you find a matching part. It's always critical to match the total number of pins when substituting parts.

One subtle detail to note in the diagram above is that some of the pins might be left unconnected. That's indicated by the simple absence of a wire connected to the pin, as in pin #16 in the example above.

You might sometimes see one (or possibly more) of the pins drawn with an "X" over it:



This means that the marked pin is meant to be snipped off on the physical pin

header, and the same pin socket in the mating connector plug is meant to be blocked (literally plugged up with a little piece of plastic, so that you couldn't insert a pin there if you wanted to). The point is to "key" the connector so that it's impossible to insert the wrong way. When you connect the plug the right way, the blocked socket in the plug lines up with the snapped-off pin on the header, so it fits and everyone's happy. If you try to insert it the wrong way, the blocked socket collides with one of the pins that wasn't snapped, preventing you from attaching it that way and alerting you that you've got something wrong.

The Pinscape schematics don't use keyed connectors like that anywhere, but it's something you might see on other schematics. The Williams pinball machines do this for most of their connectors to help prevent operators from re-connecting cables the wrong way when making repairs.

Pin numbering on the physical connector

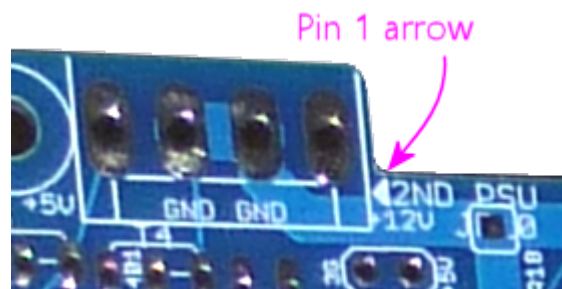
The pin numbers on the schematic symbol tell you which physical pin corresponds to each schematic pin.

Single-row pin connectors: the physical pins are numbered sequentially starting at one end. On many boards, such as the KL25Z, they indicate this by printing a "1" next to pin 1:



You can then infer all of the other pin numbers by just counting them across the row, starting at pin 1. Note that in the picture above, they've also helpfully labeled pin 3 at the other end. The KL25Z does this on each header, labeling the pins at either end.

For the Pinscape boards, the convention is to show a little triangular arrow next to pin 1:



Pin 1 is always at the end with the arrow, and the rest of pins are numbered sequentially across the row (2, 3, 4...).

Double-row pin connectors: As with the single-row headers, look for a pin 1 marking to identify pin 1. Some boards (including the KL25Z) mark this with a numeral "1" next to one of the pins. The Pinscape boards use the same triangular arrow they use for single-row connectors.

Pin 1 arrow



For double-row headers, the numbering goes by *column*:

Pin 1 arrow



71. Wire

One electrical component that you'll use a lot of in a pin cab is wire. I'd recommend buying about 100' up front if you're setting up a basic cab without feedback devices, and at least 200' if you're installing feedback devices.

For most uses in a pin cab, I'd recommend 22 AWG stranded wire. It's a good general-purpose type that will work for practically everything, from button wiring to feedback devices. For a few special cases, like power wiring and speaker wiring, I recommend 18 AWG stranded.

24 AWG works as well as 22 gauge for most purposes, and it's a little cheaper, so you can mostly use 24 if you want to save a little money. However, you'll still want some 22 AWG on hand because 24 gauge is a little thin for some of the higher-power items, particularly motors (shaker motor, gear motor, and fan). See the current-limit table below for some guidance on this. The thing I like about 22 AWG is that it has enough power capacity for just about everything in a cab, so you don't have to think too much about power limits if it's your default wire type.

Stranded or solid

Wire comes in two basic types: stranded and solid. Solid wire has a single piece of metal making up the wire. Stranded wire is made by winding several finer pieces of wire together into a bundle.

The electrical properties of stranded and solid wire are pretty similar. For a low-power application like a pin cab, there's little difference between the two types electrically. For our purposes, the main differences come down to the mechanical properties. Mechanically, stranded wire has the advantage. It's more flexible, less susceptible to metal fatigue, and works better with crimp connectors.

Copper or tinned copper

Generic hookup wire generally uses plain copper as the conductor.

There's also an upgraded type of conductor known as tinned copper, where the copper strands are coated with a thin layer of tin. The tin layer inhibits oxidation, so tinned wire is more durable, especially when the wiring might be exposed to water or high humidity. It's also more expensive; tinned copper wire costs about 2-3X as much as plain copper hookup wire.

I personally think plain copper is perfectly adequate for a pin cab, since you'll probably keep it indoors and treat it gently. That said, the pinball manufacturers deemed it worth the extra cost, because you'll find tinned wire throughout most real machines. But I think that's part of the tradition of building the real ones like tanks, which they have to do because so many are deployed in public spaces.

Gauge selection

Here are my quick rules of thumb for picking wire gauges. If you follow these, you should end up with safe wiring, without having to give a lot of thought to each individual run:

- For all feedback devices, use 22 AWG
- For button wiring and other low-power logic signals, use 24 AWG (or 22 AWG if

you prefer to minimize the different types you have to keep on hand)

- For speaker wiring and high-voltage power wiring (e.g., wires connecting power supplies to 120V), use 18 AWG

More details about wire gauge

The "gauge" or "AWG" number tells you the diameter of the wire (specifically, of the metal conductor part). The gauge scale is kind of backwards: larger AWG numbers mean thinner wire. 18 AWG is thicker than 22 AWG.

How do you choose a gauge? There are two constraints that generally bracket the size of wire you can use for a given function:

- The current carrying requirement of the function, which sets a minimum thickness
- What kind of connectors the wire will fit into

The thicker the wire (or the lower the gauge), the more current it can carry. To be really safe in terms of current capacity, you could always just use the thickest wire you can find. But thicker wire has a few downsides, too. It's harder to work with than thinner wires, and it's more expensive. What's more, there's a limit to how thick a wire will fit into a given connector. You don't want a wire that's too thick to fit a terminal that you have to connect it to.

So it's not as simple as "bigger is better". What you really want is wire that's thick enough to safely carry the current needed for its particular function, but not too much thicker than that.

If you do a little Web research, you can find lots of tables of "ampacity" - current capacity by wire gauge. Unfortunately, most of the tables have different numbers, because there are a lot of factors that go into the calculations, such as insulation type and what kind of environment the wire will be used in. For your convenience, I'm going to provide my own table below, but keep in mind that these are only approximations. I think they're pretty conservative, though, because most ampacity calculations are based on *continuous* current levels. For a pin cab, most devices are only used intermittently. Continuous usage is a much more rigorous requirement, because current limits are all about heat dissipation; a wire that only carries power intermittently doesn't have as much heat to dissipate. So for most pin cab devices, I think the numbers below represent a generous margin of safety. For devices that you plan to run continuously (light strips, for example), you might consider bumping up to the next thicker gauge if you want to be cautious.

AWG gauge	Max Amps
16	22
17	19
18	16
19	14
20	11
21	9
22	7
23	4.7
24	3.5

25	2.7
26	2.2

I highlighted the 18, 22, and 24 gauges in the table because these are the sizes I find most useful in a pin cab. You should be able to wire everything in your cab with a supply of each of these three sizes.

From the table, you can see that 22 AWG wire has a capacity of about 7A. The highest power devices in a pin cab tend to be shaker motors, gear motors, and knocker coils, and those all run at around 3 to 4 Amps. That's why I like 22 AWG as my "standard" wire type throughout the cab: it has high enough capacity to handle anything in the cab, but it's still a fairly thin wire, which makes it easy to work with and relatively inexpensive.

24 gauge has a capacity of about 3.5A, which makes it suitable for most of the "other" uses (besides heavy-duty devices like shaker motors) in a pin cab. 24 AWG is appropriate for everything carrying logic signals, like button wiring, and for low-current lighting devices like button lamps and flipper button LEDs. 24 gauge is a bit cheaper than 22, so you might prefer to use it wherever possible. But you shouldn't use it for larger mechanical devices like shaker motors and knocker coils, since those need a little more current capacity than 22 gauge can safely carry.

You can also see that 18 gauge has a very high capacity of 16A. That's plenty for the main power connections, such as the wires between the power supply and the Pinscape expansion board. It's also a good size for speaker wiring.

Labeling, color-coding and color striping

One thing you'll notice as you get into your project is that these machines use a *lot* of hookup wire. You'll be running wires to buttons and feedback devices spread around the cabinet. Many of these wires will connect to central "switchboards", particularly the key encoder and feedback controllers. That will make for a rat's nest of wires around those central points.

For maintenance purposes down the road, it's awfully nice if you can tell which wire is which at those junction points where you have lots of wires coming together. The obvious way to do that is to use a unique insulation color for each wire. But that's not really possible, because there are only about ten color choices available for common hookup wire (white, gray, black, yellow, orange, red, green, blue, purple, and brown). That's not nearly enough for all of the separate wiring functions in a pin cab. Your key encoder will have about 20 wires coming into it, and your feedback controller might have 30 to 50. You'll also have four or five different power supply connections to keep track of (ground, 5V, 12V, 24V, etc). That adds up to around 100 separate functions. With only ten colors to work with, we'll obviously have to re-use some colors for more than one function.

The place I like to start for assigning the colors is the power supply connections. These connect to practically everything in the cab, and they're especially important to keep track of, since getting them wrong can cause damage. So a consistent convention is really helpful. The main power supply lines are the "ground" or 0V (zero volts) line, which connects to practically everything, and the 5V and 12V power supply connections. The convention I like to follow here is the same one that virtually all PC power supplies use:

- Black = ground
- Red = 5V

- Yellow = 12V

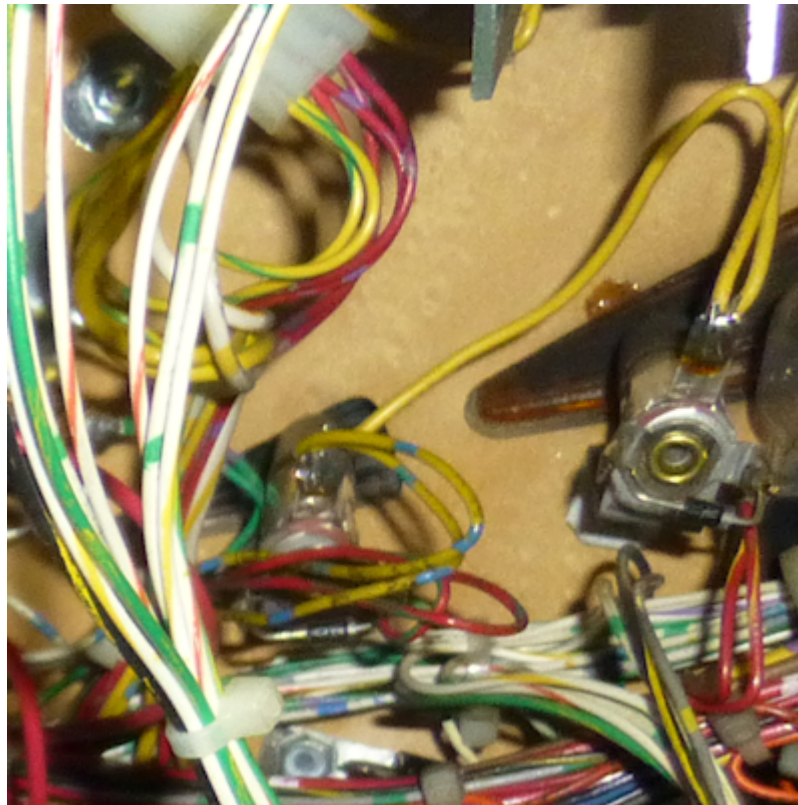
If you have a typical rainbow assortment of eight to ten insulation colors, that leaves five to seven unique colors for everything else. Which obviously isn't enough to assign a unique solid color to each of perhaps 50 to 100 separate connections to buttons, feedback devices, and speakers.

The easy way to deal with this is to reuse colors. That's what I did in my own cab. It's not ideal, but you can at least try to use separate colors for adjacent connections. For example, rotate the colors for adjacent ports on your button controller, and use different colors for two buttons situated near each other.

Reusing colors is easier during setup, but it can make future maintenance work more tedious and error-prone. If you want to be more methodical about it, and give every wire a unique appearance, for easier tracing, there are a couple of techniques available.

The first is that you can attach tags or labels to either end of each wire, with printed legends naming the function. This has the advantage that you don't have to go look at a separate chart of wire colors; you can see the intended function of each wire just by looking at the label. But I've never found wire tags to be a very satisfactory solution over the long run, because they tend to fade or fall off over time. They can also get in the way and become cluttered, especially in places where a bunch of wires come together (like the connections to a button controller).

The second approach, which is what they do in the real pinball machines, is to use "color striping". That is, you paint a stripe down the length of each wire, so that the wire isn't just "white", but "white with green stripe", say. This lets you create many unique color schemes with just a few base insulation colors.



Some examples of wires with color-striped insulation (from the bottom of a Williams playfield from the 1980s). The stripes on the white base color insulation are the most obvious, but if you look closely you can see that the dark green wire at the left is striped with yellow, and the various yellow and red wires are striped

The real machines group related connections into a common base color, to make it easier to remember the purpose of each wire. For example, a group of switches might all use green as the base color, with a different stripe color for each switch. That's a good technique to apply to your virtual cab, if you decide to use color striping.

Alas, it's not easy to buy color-striped wire off-the-shelf. Marco Specialties and Planetary Pinball Supply sell it, but I've found that they both tend to have only a few color combinations in stock at any given time, so it's difficult to set up a full collection. The other downside is that their wire runs about 2-3X what you'd pay for generic hookup wire, because the type they sell uses tinned copper (see above). You can find striped wire from a few other Web vendors as well, but the ones I've been able to find have very limited color and gauge selections.

If you want to use striped wire, you might actually be better off creating your own rather than trying to buy it. There's a clever DIY system for creating your own striped wire out of the cheaper solid-color wire. Some of the home-brew pinball people use this. (Yes, there really is such a thing as home-brew pinball - people who build *physical* pinball machines of their own design from scratch.) Here's a page on the subject on one of the DIY pinball sites:

Color coding - pinballmakers.com

Their approach is as follows:

- Start with a collection of ordinary hookup wire in assorted solid colors. These will serve as the base colors.
- Get a set of **oil-based** Sharpies or similar permanent markers in assorted colors. These will be used for the stripe colors. Oil-based inks are a must for this - regular water-based inks won't adhere to the plastic insulation. (I've tried it; the ink will just end up all over your hands.)
- As you dispense wire, use one of the oil-based markers to stripe it.
- To make the striping more automatic, set up a dispenser that feeds the wire through a slot that the marker sits on top of. The pinballmakers site suggests using a PVC T joint to hold the pen.

The great thing about this approach is that you don't have to buy a hundred spools of wire with unique color combos. You just need a basic rainbow assortment of the common single-insulation-color wire, plus a small assortment of markers. And it doesn't add a lot of extra prep work, since you create the color striping as you spool out the wire.

72. Resistors

The resistor is one of the simplest component types. A resistor's function is (as the name suggests) to resist electric current. Electrical resistance is somewhat analogous to friction, in that resistance sloughs off some of the electrical energy going through a circuit and converts it to heat.



Technically, just about everything in a circuit has some resistance, even the wires, although that's all usually small enough that we mostly ignore it. Resistors usually have much higher resistance than what's naturally there from the wires and the rest. Resistors are useful because they let us insert a specific amount of resistance into a circuit just where we need it, to control the way current flows.

Most resistors look something like the ones pictures at right: a little roughly-cylindrical main body with a wire coming out of each end. (It's usually more of a dumbbell shape than a true cylinder, but close enough.) This main body can range in length from a few millimeters to about an inch. The body is marked with colored stripes that use a secret code to indicate the resistance value. If you buy your resistors from someone like Mouser that packages each component in a clearly marked plastic bag, you probably won't have to bother learning to read the color code, but it's explained below in case you're interested.

Orientation

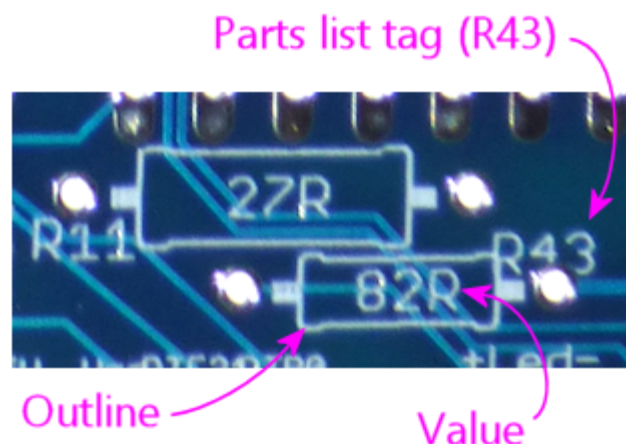
Resistors aren't polarized, meaning that they don't have a preferred direction. You can install a resistor in a circuit in either direction and it'll work the same way.

Installing in a circuit board

All of the resistors on the Pinscape board are the same shape: little cylinders with wire leads sticking out of each end. Most of the ones on the parts list are rather tiny "thin film" resistors, most only a a few millimeters in length.



The install location for a resistor on the circuit board is marked with a little outline of the resistor showing how it'll look when installed, when viewed from straight overhead. This shows an outline of the barrel-shaped body, with lines on either side representing the wire leads.



(82R = 82 Ohms)

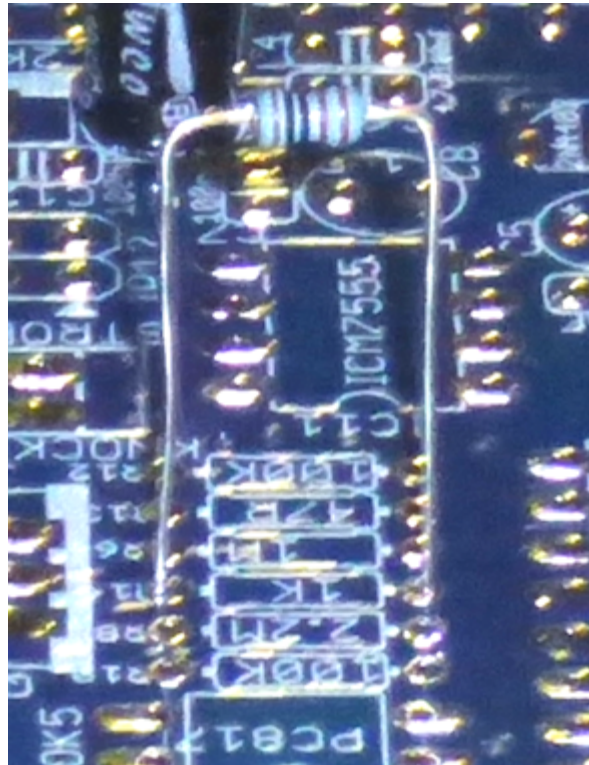
The lines sticking out of the body-shape outline extend to the solder pads where the resistor is installed.

A resistor's wire leads will usually start out sticking straight out from the resistor body. You have to bend the leads at a 90° angle on either side of the body to fit it into the board, like this:



Use the spacing of the holes as a guide to where to make the bends. The holes are usually placed so that you simply have to bend the leads as close to the resistor body as you comfortably can. You can just bend the leads by hand, but you'll get sharper corners if you use needle-nose pliers, which can make the resistor fit into the board more easily.

Once the leads are bent to fit the hole spacing in the board, fit the ends of the leads into holes, and feed them all the way through until the resistor is seated more or less flat against the board.





As always, there's no need to force anything - just get the resistor as close to flat as you comfortably can. The fit is sometimes a little too tight to get it perfectly flat, which is okay.

Once the part is in place, hold it there, flip the board over, and solder the leads to the pads on the bottom of the board. Snip off excess length from the leads after the solder cools.

On schematics

Resistors are represented on schematics as a jagged lines, as shown at right. They're usually marked with an **Rnumber** reference designator, such as R5 or R10. The "R" is for resistor, and the number is arbitrary, purely to match up against the parts list. Designators always have to be unique within the whole schematic, so that each part can be individually identified. Resistors on schematics are also usually labeled with the resistance value in Ohms, which we'll come to shortly. In this case, the "4.7K" label means 4.7 Kilo Ohms = 4,700 Ohms.



Somewhat confusingly, the symbol "R" is *also* used as a substitute for the Ω symbol for Ohms, so you might see a resistor with the pair of labels "R6" and "4R7". The one with the number *after* the R is the part label, so this is part number R6. The one with the number *before* the R is the Ohms value. In this case, we have a number both before and after the "R", which you should read by replacing the "R" with a decimal point and using the result as a number of Ohms: so "4R7" means 4.7 Ohms. In my own schematics and boards, I prefer to keep all of the digits before the "R" and simply use regular decimal points, so I'd write 4.7R in this case. But you might see the "4R7" type notation in other people's schematics and boards. The same convention applies to the K (for Kilo Ω) and M (for Mega Ω) symbols, so you might see 4K7 instead of 4.7K, or 1M2 instead of 1.2M.

Here's a summary of the common formats:

- **47R** means **47 Ohms**
- **4R7** means **4.7 Ohms**
- **47K** means **47 Kilo Ohms** = 47 k Ω = 47000 Ohms
- **4K7** means **4.7 Kilo Ohms** = 4.7 k Ω = 4700 Ohms
- **47M** means **47 Mega Ohms** = 47 M Ω = 47,000,000 Ohms
- **4M7** means, you guessed it, **4.7 Mega Ohms** = 4.7 M Ω = 4,700,000 Ohm

Selection

The main parameter for selecting a resistor is the resistance in Ohms. In some cases, it's also necessary to pick a resistor with a high enough power handling capacity in Watts.

Resistance value (Ohms): Every resistor has a numerical value specifying how much electrical resistance it has. The resistance is marked on most resistors via the colored stripes we just mentioned; in some cases, it's simply printed numerically instead. Either way, the value is in Ohms, which has the symbol Ω : so 100 Ohms can be written as 100 Ω . Thousands of Ohms are written as Kilo Ohms, K Ohms, K Ω , or simply K, so 4.7K means 4,700 Ohms. Similarly, 1M Ω = 1M = 1 Mega Ohm =

1,000,000 Ohms.

Pay close attention to the "K" or "M", and especially pay attention to the **absence** of a "K" or "M". One of the most common mistakes that people make when reading schematics or parts lists is to assume that resistor values are always in "Kilo Ohms" even when they don't see a "K" in the Ohms value. This especially happens with LED resistors, which are sometimes really tiny values like 4.7 ohms. A lot of people see that "4.7" and think that there was supposed to be a "K" there because there's *always* a "K" there! They think maybe the "K" was left out by accident, or it's one of those conventions where engineers leave out some implied information because everyone knows it's always there whether it's stated or not. There really are a lot of situations like that in engineering, but this isn't one of them! If there's no "K" stated, there really is no "K". "4.7 Ohms" is a perfectly valid resistor value, and it's very, very different from "4.7 K Ohms". So always pay attention to whether or not a "K" is there.

You might sometimes see a different notation in schematic drawings and on printed circuit boards. In these contexts, it's become conventional to replace the symbol Ω with the letter R, in part because of the difficulty of writing Ω on older computer systems. If you see 100R, it means the same thing as 100 Ω . You might see an even stranger notation, with numbers on both sides of the "R", as in "6R8". In this case, replace the R with a decimal point: so 6R8 means 6.8 Ω . Similarly, 4K7 means 4.7 K Ω . There's a method to the madness here: they use this notation because decimal points aren't always legible in schematics, and can be especially impossible to see in the tiny text silk-screened onto circuit boards.

Wattage: Resistors have a second rating, separate from the resistance value, specifying how much power they can handle. This is given in Watts. Typical resistors are rated for either 1/8W or 1/4W. For higher wattage values, you have to find parts specially made for the higher power.

There are no markings on a resistor's physical package to tell you the wattage value (the way the color stripes tell you the Ohms value). The only way to find the wattage is to refer to the package the resistor came in. When you're buying resistors, though, it's one of the parameters you can use to select parts.

In the parts lists in this guide, we'll always tell you if you need a specific wattage. In most cases, any wattage rating will work, because in most cases the actual power usage will be less than 1/8 W, which is almost always the smallest wattage value that they make resistors in at all. If we don't say anything about wattage, you can go by the Ohms value alone and choose a resistor with any wattage rating. In cases where a specific wattage **is** required in a parts list, it's always a **minimum**. You can use any resistor rated for that wattage or higher. For example, if a parts list calls for a 1/2W resistor, you can use a resistor with a 1/2W rating, or a 1W rating, or anything higher. But do be aware that a higher wattage rating usually means the resistor is physically larger, so don't go overboard. If you're going to use a resistor in a circuit board, you don't want to get something so huge it won't fit in its allotted space.

Stripe color code

Most resistors are marked with a set of color stripes that tell you the resistance value of the part, using a special color code system.

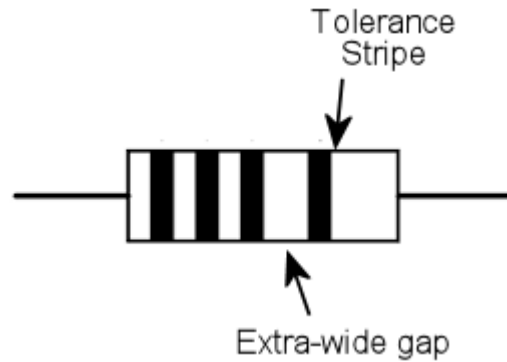
Three, four, or five stripes

Color-coded resistors are marked with three, four, or five stripes.

If there are **three stripes**, all three stripes represent the resistance value in ohms,

as explained below.

If there are **four stripes**, three of them should be grouped together and one should be set apart by an extra-wide gap. The three grouped stripes represent the Ohms value, exactly like a three-stripe resistor, and the separate fourth stripe represents the "tolerance" value. Both are described below.

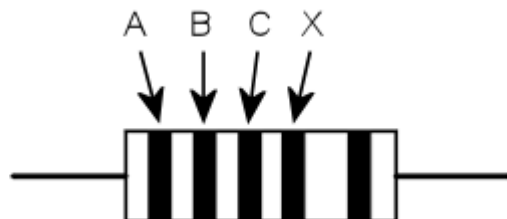
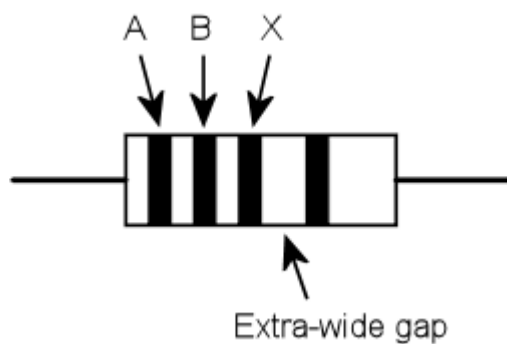


If the resistor has **five stripes**, the first four represent the Ohms value, and the fifth represents the tolerance.

Ohms Value

The Ohms value of a resistor can be read from the group of three stripes in a three- or four-stripe resistor, or the group of four stripes in a five-stripe resistor.

The first step is to read the colors, starting with the one nearest the end. If you've already found the "tolerance" stripe, start at the opposite end from that. Otherwise, start at the stripe closest to an end.



The next step is to convert the colors to digits, using this table:

Color	Digit	Multiplier (10^X)
Black	0	1
Brown	1	10
Red	2	100
Orange	3	1,000

Yellow		4	10,000
Green		5	100,000
Blue		6	1,000,000
Violet		7	10,000,000
Gray		8	100,000,000
White		9	1,000,000,000
Gold			0.1
Silver			0.01

Finally, string the digits together into a number, like so:

AB times 10^X or ABC times 10^X

As you can see, the last stripe represents a power-of-ten multiplier. You can read it as 10^X , using the same color-to-digit mapping to get the X value. Or you can just use the "multiplier" value in the table above, which we wrote out for each power of ten. It's the same value either way.

You might notice that gold and silver are special cases. They represent 10^{-1} (0.1) and 10^{-2} (0.01) respectively. You'll never see them in a "digit" slot - only as a multiplier.

For example, for Yellow-Violet-Red, you'd read this as A=Yellow=4, B=Violet=7, X=Red=2, so you'd form the numeric value as

$$\text{AB times } 10^X = 47 \text{ times } 10^2 = 47 \times 100 = 4700$$

That means you have a 4700 Ohm resistor, usually written as 4.7K Ω or just 4.7K. 1K Ω = 1000 Ω , and 1M Ω = 1,000,000 Ω .

Tolerance (precision)

If the resistor has four stripes, one of them, always at one end, is the "precision" or "tolerance" stripe. If the resistor has five stripes, the fifth is the precision stripe. This extra stripe should be separated from the other stripes by an extra-wide gap, to visually identify it as standing apart from the others.

This extra stripe tells you how carefully calibrated the resistor is. Some situations call for very precisely calibrated resistors, others only need something approximate. It's more expensive to manufacture parts with better calibration, so the manufacturers offer cheaper, less precise parts for when you don't care, and pricier, better calibrated parts for when you do. This stripe tells you what sort you have in hand.

The tolerance is expressed as a plus-or-minus percentage value. A tolerance of $\pm 10\%$ means that the resistance of the part is within 10% of the resistor's nominal value, above or below. For example, a 100 Ohm resistor with a 10% tolerance (silver stripe) should have an actual resistance value within 10 Ohms of the nominal 100 Ohms, meaning the actual resistance should be from 90 Ohms to 110 Ohms.

If there are only three stripes, the tolerance is implicitly 20%.

Here are the code colors:

- Brown = $\pm 1\%$
- Red = $\pm 2\%$
- Green = $\pm 0.5\%$

- Blue = $\pm 0.25\%$
- Violet = $\pm 0.1\%$
- Gray = $\pm 0.05\%$
- Gold = $\pm 5\%$
- Silver = $\pm 10\%$

73. Capacitors

The capacitor is a basic electronic building block that adds a measured amount of "capacitance" to a circuit, or the ability to temporarily store a quantity of electric charge.

The ability to store charge makes capacitors look glancingly similar to batteries, but they're rather different in their physical construction and electronic properties, and they're used for different purposes. A battery uses a chemical reaction to store energy, which allows it to store potentially large amounts of energy over long time spans. In contrast, a capacitor stores its energy in a standing electric field, so it can only store a (relatively) small amount of energy over a short period of time. This makes a capacitor a poor power source. Instead, the property of a capacitor that's most often used in circuit design is its ability to resist changes in applied voltage, which happens because the stored electric field acts as a sort of buffer that has to be charged up or depleted before voltage changes can get past the capacitor. This effect can be exploited for purposes such as signal frequency filtering and power line conditioning.

Types of capacitors

All of the capacitors used in the Pinscape projects fall into one of two categories: ceramic disc capacitors and electrolytic capacitors. These aren't the only types that exist - there are about a dozen in all - but they're by far the most common in everyday electronics. Most other types are only seen in specialized applications.

Ceramic disc capacitors



These capacitors are usually flat and disc-shaped, but sometimes they're flat and squarish. Small ones can be bulbous or cigar-shaped.

Ceramic capacitors aren't polarized, so they can be installed in either direction. There's no "+" or "-" leg.

Disc capacitors tend to have small capacitance values, usually below 1 μF (micro Farad). Common sizes are measured in fractions of a μF , such as 0.1 μF , or in whole nF (nano Farads). Very small ones are measured in pF (pico Farads).

It's best to keep disc capacitors with their packaging until you're ready to use them, because they can be so tiny that it's difficult to read any markings. If you need to identify a disc capacitor, look for a three-digit number printed on the face of the disc.





For a three-digit number, take the first two digits, and add the number of zeroes given by the third digit. So for the "154" example pictured above, we'd take out the "15" and add 4 zeroes, giving us 150000. This is always a value in pF units - pico Farads, equal to trillionths (10^{-12}) of a Farad. 1000 pF is the same as 1 nF, so our "154" capacitor above is 150000 pF or 150 nF, which is also the same as 0.15 μ F.

If a letter follows the three-digit number, it's a tolerance code telling you how precise this measurement is. This is typically K for $\pm 10\%$ or M for $\pm 20\%$. (You can find tables of these codes online if you need the full set; search for "disc capacitor code".)

Electrolytic capacitors



Electrolytic capacitors usually come in cylindrical metallic cases as pictured above. The leads may be "radial" as shown (both sticking out the same end), or "axial" (each lead sticking out one end of the cylinder).

These capacitors are polarized, meaning that they're sensitive to the direction of the voltage applied. One leg has to be connected to the "+" voltage and the other to the "-" voltage. See the section below on orientation for help figuring out which leg is which.

Most electrolytic capacitors have relatively large capacitance values, measured in whole μ F (micro Farads). The smallest electrolytics are around 1 μ F, and the largest are very large indeed, sometimes in the hundreds of thousands of μ F. (Very large ones might even be measured in mF - milli Farads - although it seems more common to stick with the μ F units even for very large values.) For the Pinscape projects, the largest ones you'll see are around 1000 μ F, which run about an inch tall and half an inch in diameter.

The capacitance value of an electrolytic is usually printed right on the case directly, including units, as these capacitors are big enough physically that there's no need for secret codes. You might see a label saying "1000 μ F", for example.

Installing in a circuit board

Most of the capacitors used in the Pinscape project use radial through-hole leads, meaning that they have two wires sticking out their bodies, side by side.

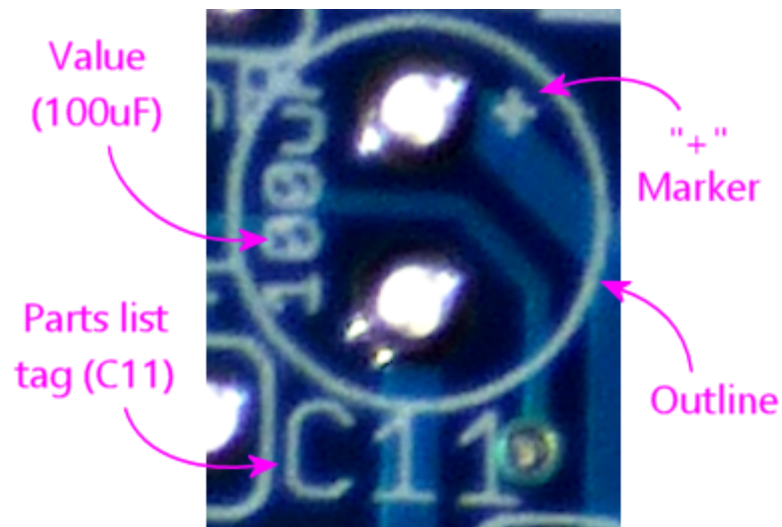
Find the marked spot on the circuit board for the capacitor. It should show an outline

that roughly matches "footprint" of the part when installed. Then fit the leads through the holes in the solder pads. Feed the leads all the way through until the part is nearly flush with the board.

For example, a ceramic disc capacitor is typically shown on the circuit board with a little rectangular outline:



An electrolytic is shown on the circuit board with a circular outline, with one solder pad marked with a "+" sign to indicate which way to orient the part:

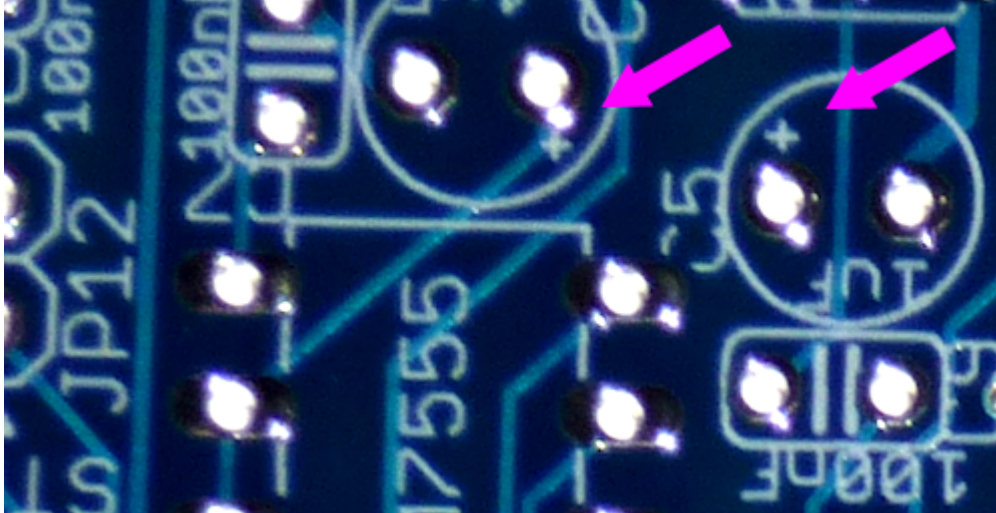


Orientation

Ceramic disc capacitors are unpolarized, meaning that they don't have a preferred direction. You can install these in a circuit in either direction.

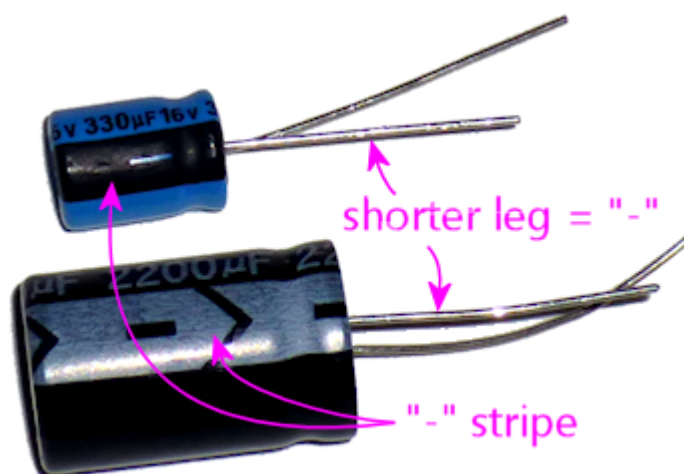
Electrolytic capacitors, in contrast, are polarized. They must be installed in the correct direction. To determine the correct direction to install one on a circuit board, find the part outline printed on the board, and look for a little "+" sign printed next to one of the solder pads:



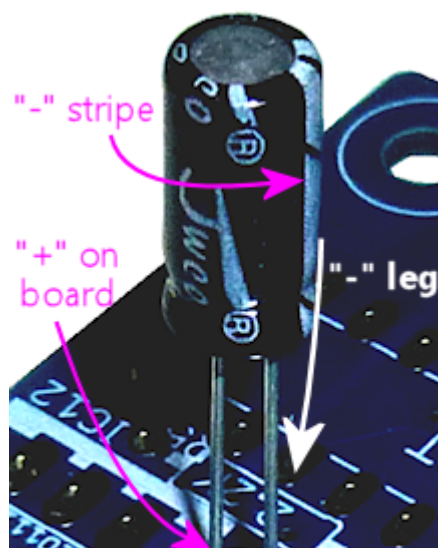


This is the pad where you solder the "+" leg of the capacitor, so the other, unmarked pad is for the "-" leg.

Now look at the capacitor itself. There should be a prominent stripe painted along one side of the barrel. In most cases, this will be marked with "-" (minus) signs. The leg closest to this stripe is the "-" leg of the capacitor. In addition, the legs will usually be different lengths: the short leg is "-".



Match up the leg on the capacitor near the "-" stripe with the unmarked "-" solder pad on the board, and match up the unmarked "+" leg of the capacitor with the marked "+" solder pad on the board.

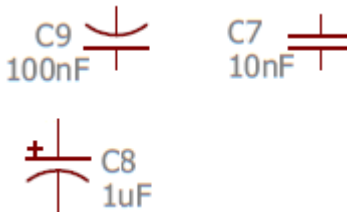




As with most components, it's best to seat the capacitor close to the board. There's no need to force anything, but get it as close as you comfortably can. Solder the leads on the bottom side of the board, and snip off the excess after the solder cools.

On schematics

Capacitors on schematics are shown with symbols that look like two parallel lines, or one straight line and one slightly curved line.



For electrolytic capacitors, which are polarized and thus have "+" and "-" ends, the schematic symbol will show a little "+" sign on the positive end. If there isn't a "+" sign, it means that an unpolarized capacitor, such as a ceramic disc type, has to used here.

Capacitors are usually marked with a **Cnumber** designator, such as C7 or C9. The "C" is for Capacitor, and the number is arbitrary, purely to serve as a reference to the parts list. One of the cardinal rules of schematic writing is that designators have to be unique in the schematic, so that every part can be individually identified.

The schematic will usually also show a capacitance value in Farads (F). This will almost always use one of the following formats:

- **100pF** means **100 pico Farads** or 100 trillionths of a Farad
- **100nF** means **100 nano Farads** or 100 billionths of a Farad
- **100uF** means **100 micro Farads** or 100 millionths of a Farad

Selection

Capacitors have three main specs: the type, the capacitance value in Farads, and the maximum voltage rating.

Type: In most cases, either **ceramic disc** or **electrolytic**. There are about a dozen other more exotic types (glass, air-gap, film), but ceramic disc and electrolytic are by far the most common. They're the only types needed for the Pinscape projects.

You always have to match the type shown in the schematics and parts list. If it's not clear from the parts list, check the schematic symbol for the capacitor: if it includes a "+" sign, an electrolytic type is needed, otherwise a ceramic disc type must be used.

Capacitance: A value in Farads, almost invariably expressed in μF , nF, or pF (micro, nano, or pico Farads). Match the value specified in the parts list or schematic exactly when selecting a capacitor.

Pay attention to the units - μF , nF, and pF are very different! 1 μF is 1000 nF, and 1 nF is 1000 pF. If you substitute a 1nF capacitor for one that was supposed to be 1pF, you'll be off by a factor of 1000; if you substitute 1 μF for 1pF, the error is a factor of a million!

But by the same token, you can take advantage of the factor-of-1000 relationships to figure equivalences. If you're looking for a 0.1 μF capacitor, you can substitute 100nF, since it's exactly the same value expressed with a different multiplier.

Voltage rating: Every capacitor is rated for its maximum allowable voltage. This is the highest voltage that it can be exposed to in the circuit.

If the parts list specifies a voltage rating (e.g., "100 μF /50V"), you must select a capacitor rated for *at least* that voltage. So if the parts list says you need a 50V capacitor, you can use a 50V capacitor, or a 100V capacitor, or anything higher.

If the parts list doesn't specify a voltage rating, it means that the lowest rated available capacitors (usually 25V) can be used.

Precision: Capacitors are also rated for precision, also known as tolerance. This is usually given as a percentage, typically 10% or 20%. This means that the manufacturer claims the part will be within the stated range of its nominal capacitance value. For some applications, it's critical to be very close to a particular value, so the engineer who designed the circuit might specify that you need a 5% or 1% tolerance capacitor in a particular spot. The Pinscape projects don't have any such requirements, so you don't have to worry about the tolerance value when selecting parts for these boards.

74. Diodes

The diode is perhaps the most fundamental semiconductor component. It serves as a one-way valve in a circuit, allowing current to flow in only one direction.

If you've looked at installing any tactile feedback devices for your system, such as solenoids or motors, you've probably already run into one common use of diodes in pin cabs, which is to block the harmful "flyback" current that coils produce when switching off (see Chapter 53, Coil Diodes). Diodes have many other uses as well, such as voltage rectification (converting AC current to DC) in power supplies, and as components in logic circuits.

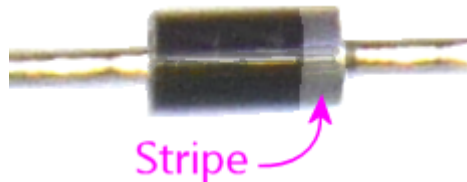
Physically, a diode typically looks like a small black cylindrical body with wire leads sticking out of each end.



Orientation

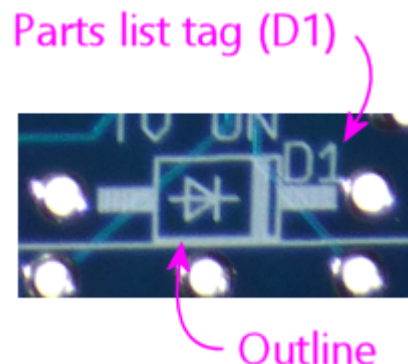
Diodes are polarized (they have a "+" end and a "-" end), so it's important to orient them correctly when installing in a circuit.

The standard convention is that the "-" end of the diode is marked with a painted stripe - a ring around the cylinder. It's usually a white stripe to contrast with the black case, although sometimes the ink is so thinly applied that it comes out a pale gray. You might need a bright light to see it. I had to artificially increase the contrast in these photos to make the stripe apparent.



Installing in a circuit board

The circuit board install location for a diode is marked with a little outline of the diode as seen from above when installed, showing the cylindrical body, and lines extending on either side to represent the wire leads. The lines extend to the solder pads where the diode is to be installed.

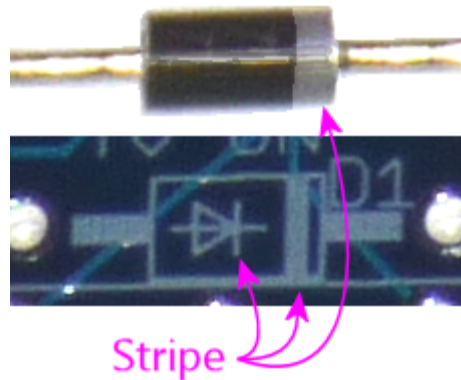


This looks very similar to the outline for a resistor, so to help clarify that it's a diode,

the outline has the schematic symbol for a diode inside:



This symbol is also there as an orientation marker, so pay close attention to it! The stripe on the diode goes on the side with the stripe in the picture:

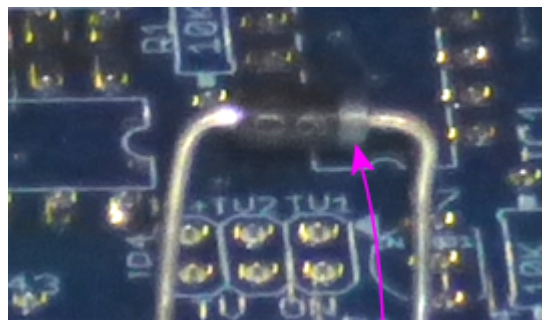


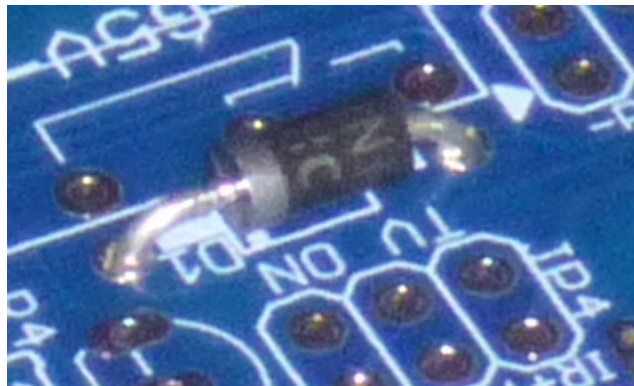
As with resistors, a diode's wire leads usually start out sticking straight out from the body, so you have to bend each lead at a 90° angle to fit it through the closely-spaced holes in the circuit board. You can bend the leads by hand, or use needle-nose pliers if you want a sharper corner, which can make for an easier fit on the board. Use the spacing of the holes on the circuit board to determine where to bend the leads - usually they have to bent as close to the body as you can get.



With the leads bent to fit the hole spacing in the circuit board, fit the ends of the leads into the holes in the solder pads, then feed the leads through until the diode is seated flat against the board. As always, you should get it as close as comfortably possible, but don't force it.

Be sure to orient the diode properly! Remember, the stripe painted on the aligns with the stripe in the circuit board outline, and also with the "bar" in the diode symbol printed on the board.





With the diode seated - and after you've double-check the orientation! - hold the diode in place, flip the board over, and solder the leads to the pads on the bottom side of the board. Snip the excess length from the leads after the solder cools.

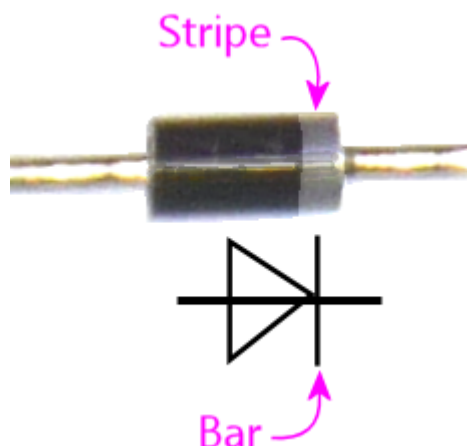
On schematics

A diode is represented by a symbol like this on a schematic:



This is meant to represent the diode's one-way-valve property; the arrow points from positive to negative, and the little bar indicates the barrier to reverse flow.

When relating the schematic symbol to an actual diode, the key is to line up the bar on the schematic symbol with the stripe on the diode body.



The diode symbol on a schematic should always be accompanied by a reference designator, usually a "D" number (D1, D2, etc), "D" for diode. As always, this is just an arbitrary name for looking up in the parts list and doesn't mean anything by itself.

You'll also usually see (as we do here) a part number for the diode, in this case 1N4007. Many diodes have generic "1N" numbers, indicating a common type of diode that's made by multiple manufacturers to the same specs.

(If you've taken a high school physics class, you probably know the jaded history of this symbol and how it points in the "wrong" direction in a physical sense. During the early days of electronics, it was believed that the charge carriers in an electronic circuit were positively charged ions, so the convention was to draw diagrams showing current flowing from positive to negative. It was later discovered that the actual moving particles are electrons, but everyone agreed it would be too confusing to change, given all of the existing literature based on positive-to-negative current flow. So we still use those original conventions to this day, even though everyone now understands that the electrons are actually traveling *against* the arrow. Engineers today just make the reversal mentally whenever they have to think about the physics of it. At least they're consistent: the same wrong-way-arrow can be found throughout electronics symbolism, such as in the symbols for transistors and MOSFETs.)

Selection

Unlike resistors and capacitors, diodes don't have any simple "unit" that describes them. So instead, schematics and parts list will specify a particular diode to use, by part number.

Part numbers starting with "1N" are common, generic types of diodes that are made by multiple manufacturers to the same specs. There are also less common diodes manufactured for specific properties by one company, so you might see specific manufacturer part numbers instead of "1N" numbers as well.

In any case, use the part number to search at Mouser or another vendor to find matching parts. It's always easiest to go with an exact match, but in some cases, you might find cross-references where the manufacturer says that a different part can be substituted for the part you're looking for. These substitutions are usually listed in tables in the data sheets, so searches on vendor sites might not turn them up, but Google searches might. You can try search terms like "1N4007 equivalent" if you have trouble locating the exact part you're looking for.

75. LEDs

LEDs are in fact diodes - it's right in the name, Light Emitting Diode. They're just a special type of diode that glows when current is flowing.

Using an LED in a circuit board is mostly like using any other diode in a circuit board, so you might want to take a look at Chapter 74, Diodes. The main difference with an LED is that the physical package is usually a little different. Through-hole LEDs usually use "radial" packages, with two leads sticking out of the bottom.



Static electricity warning

LEDs are sensitive to static electricity. Refer to Chapter 67, Static Electricity Precautions for tips on handling static-sensitive parts.

Identifying the positive and negative legs

Regular diodes use a stripe on the body to indicate the negative end, but that doesn't work for most LEDs because of the different package shape. Instead, LEDs usually indicate the positive and negative sides using a longer leg and a shorter leg:

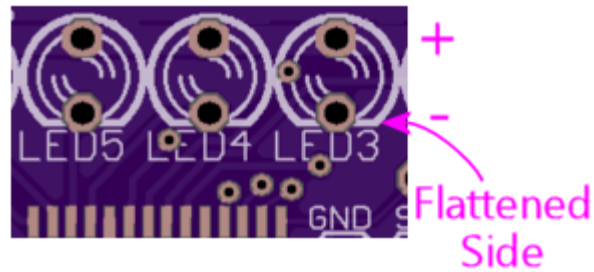


It's actually kind of mnemonic: shorter = minus.

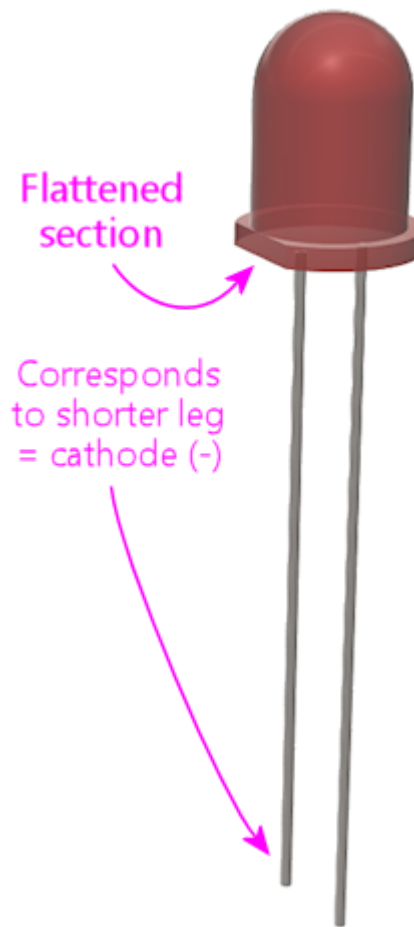
The negative lead of an LED is also known as the cathode; the positive lead is the anode. (Here's my handy trick for remembering which is which. Remember CRT TVs? CRT stands for Cathode Ray Tube. We know that a "Cathode Ray" is actually an electron, and that electrons have negative charge. So the cathode is the negative side.)

Orienting on a circuit board

The circuit board markings for an LED will indicate the orientation, either by marking one of the pads with a "+" sign for the "+" leg, or via the outline on the circuit board. If it's indicated via the outline, one side of the outline circle will be flattened:



The flat side on the diagram corresponds to the flat side on the base of the LED's plastic bubble. It's pretty subtle on smaller parts - so much so that I can't get a good photo. Instead, here's a 3D rendering that exaggerates it enough to see clearly:



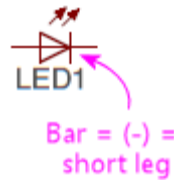
To orient the LED for installation, line up the flat side on the base of the bubble with the flat side of the outline on the circuit board. If there's not an obvious flat side, match up the short leg on the LED with the flattened side on the circuit board outline.

On schematics

The schematic symbol for an LED reflects that it's really a diode, by using the same basic symbol. But there's also a little embellishment added to show the special light-emitting feature: a couple of little arrows pointing out, representing the photons being emitted.



As with a regular LED, the arrow points from positive to negative, so the side with the "bar" is the negative lead or cathode.



LED specs

LEDs are characterized mainly by two values: the **forward voltage** and **forward current**.

A lot of people get confused about the forward voltage number, thinking that it means the required supply voltage. That's a completely reasonable mistake, because for just about any other kind of device, a "voltage" spec would mean just that. But for an LED, it's something different. The forward voltage is actually the voltage that the LED "drops" in the circuit, which is sort of like consuming the voltage. In order for an LED to function, it must be supplied with *more than* its forward voltage.

For example, if you have an LED rated with a forward voltage of 3.5V, it'll work if you supply it with 5V, since 5V is greater than the rated forward voltage. It *won't* work if you supply it with 3V, because that's less than the rated forward voltage.

The supply voltage has to be higher than the LED's forward voltage, but it's more efficient if it's not *too* much higher. The higher the voltage, the more power has to be wasted in current-limiting resistors. For example, if you have an LED with a 3.5V forward voltage, and you have a choice of powering it with a 5V or a 12V supply, you should choose the 5V supply. Both 5V and 12V are above the required threshold, so either one will work, but 5V is more efficient because it's the lower voltage option.

You often see LEDs rated with a range for the forward voltage, such as "3.2V - 3.4V". That just means that there are some variations in the manufacturing process, so individual LEDs will each be a little different, but each one should be somewhere in that range. If you're trying to figure the required supply voltage, just make sure the voltage is higher than the high end of the range.

The other spec for an LED is the forward current. That's more straightforward: that actually is the current level that you're supposed to use with the LED. You might also see a separate maximum forward current; this is a level you should never exceed. LEDs don't vary much in brightness when you change the current level, but they do get hotter, so there's usually no benefit in using a higher current than the recommended forward current. Doing so just reduces the lifetime of the part.

Selection

LEDs are usually placed in a circuit solely for their ability to emit light, as opposed to serving some kind of active function that affects the other electronics in the circuit. This makes them more interchangeable than regular diodes, where the circuit designer might be relying on a particular diode's special electronic characteristics.

So in most cases, the only two things that matter when selecting an LED are its **forward voltage** and **forward current** ratings. And you don't typically have to match these exactly - you just need compatibility. What constitutes compatibility depends on the circuit:

- If the circuit has active current regulation, so that it won't exceed a given current level through the LED, an LED is compatible with the circuit if the LED's forward voltage is *lower* than the power supply voltage in the circuit, *and* the LED's maximum forward current rating is at least the regulated current level that the circuit uses.

For example, if your circuit board provides 5V power to an LED with current regulated to 20mA, you can use an LED with a forward voltage of 2V (because 2V is less than the 5V supply voltage) and a maximum current rating of 50mA (because 50mA is greater than the regulated 20mA). You *can't* use an LED with a forward voltage of 6V, because that exceeds the 5V supply voltage.

- If the circuit board uses a resistor to regulate the current (see Chapter 52, LED Resistors, and you *can't* substitute a different resistor, you can use any LED with approximately the same forward voltage that the circuit designer specified, and with a maximum current rating at least as high as the forward current that the circuit designer specified. I can't give you a hard-and-fast rule for exactly how "approximate" you can be, but let's say within 10% or so.

For example, if your circuit board uses a resistor to regulate current and calls for an LED with a 3.2V forward voltage and 20mA forward current, you can use any LED with a forward voltage of about 2.9V to 3.5V (within 10% of the specified 3.2V), and a maximum forward current of 20mA or higher.

- If the circuit board uses a resistor to regulate the current, and you *can* substitute your own resistor in place of what the designer specified, you can use any LED with a forward voltage less than the supply voltage in the circuit.

In this case, you'll have to select an appropriate resistor based on the LED you actually end up using. See Chapter 52, LED Resistors for the formula for selecting the resistor. When figuring the resistor value, use the LED's forward voltage spec, and use the desired current in the circuit. The desired current must be less than or equal to the maximum current specified for the LED, and it must also be less than or equal to the maximum current that the circuit can safely supply. For that, you'll have to consult the circuit board's specifications or ask the designer.

If you're installing an LED in a circuit board, it also obviously has to match the physical size and shape that the board was designed for.

76. Relays

A relay is a mechanical switch controlled by an electromagnet, allowing one circuit to control another circuit. Because the switching is done mechanically, there's no need for the two circuits to be connected electrically, which simplifies control interfaces between incompatible circuit types. One very common use along these lines is controlling a high-voltage AC power line circuit from a low-power DC logic circuit.

The Pinscape main expansion board includes a relay for the TV switching function. It uses a relay precisely because of the isolation the relay provides: it doesn't require any electrical contact between the TV and the Pinscape boards, so it doesn't have to make any assumptions about how the TV control switches are wired.

This chapter is about relays as electronic components in general. There are a couple of specific ways that relays are commonly used in pin cabs that we cover separately in other sections:

- Relays can be used as an interface between the popular LedWiz feedback controller and higher-power devices such as solenoids and motors. The LedWiz can't be directly connected to large devices because of its relatively low power handling limits. Relays can be used to bridge that gap, although there are better alternatives, such as MOSFETs. This is covered in Chapter 50, LedWiz Setup.
- Some pin cab builders use USB-controlled relay boards from Sainsmart as their output controllers. These are turn-key controllers that let you switch a bank of mechanical relays via USB commands, so they can be used for many of the same functions as an LedWiz. They're covered in Chapter 51, SainSmart Relay Board Setup.

Protective diodes

The physical mechanism inside a relay that operates its switch is an electromagnet. This is a type of electromagnetic coil, which means that it's affected by a phenomenon known as "flyback current" that affects all coils. Flyback current is essentially a type of electrical interference or noise that gets injected into the circuitry attached to the relay coil, and this interference can be harmful to other components in the same circuit.

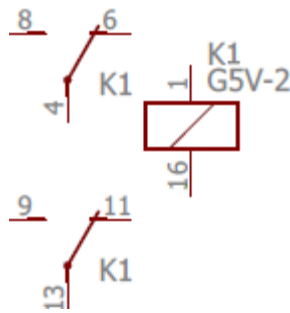
There's a standard remedy to protect against this harmful interference, which is to connect a diode to each coil. We cover this in detail in Chapter 53, Coil Diodes.

We wanted to call your attention to it here because it might not be obvious from reading that section that a relay is one of these coil-based devices. That section is all about things like motors and solenoids. But a relay is in the same class because of its integrated electromagnet. Note that if you're controlling a coil-based device using a relay, such as a motor or solenoid, that device will need its own protective diode, in addition to the one needed for the relay coil. Also note that it's the relay *coil* that requires the diode, not the relay *switch*. The coil is the part where the "input" or control signal feeds into the relay.

The Pinscape boards already include the necessary protective diodes for their relays, so you don't have to worry about adding your own separate diode there. All of the Sainsmart boards should likewise include pre-installed diodes.

On schematics

There are several different ways to represent relays in schematics. Here's the format that we use in the EAGLE plans for the Pinscape boards:



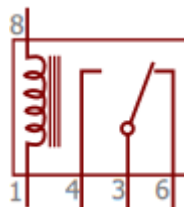
This breaks up the relay into its component parts:

- the little box at the right represents the electromagnet
- the two clusters at the left represent the mechanical switches (this particular relay has two of them, because it's a "double pole" relay, meaning it has two electrically independent switches operated by the same electromagnet)

You can tell that all of these little sub-symbols are actually part of one physical device from the repeated reference designator: all three parts are labeled "K1". This convention (of breaking up one part into multiple schematic symbols) is a little subtle, but it's pretty common; it's explained in more detail in "Multi-gang chips" in Chapter 70, Schematics.

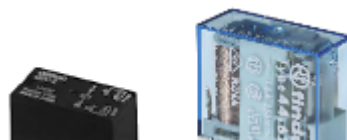
The numbers on the connection points indicate the pin numbers on the physical relay. These numbers aren't anything you're supposed to "just know" on your own - they come from the relay's data sheet. Every relay uses its own peculiar numbering, so you just have to look it up for each relay.

The Pinscape EAGLE schematics use the symbology above for relays, but it's not the only one. If you look at other schematics, you might see other relay symbols. The most common format is a box containing a little pictogram of a coil representing the electromagnet, and one or more switch symbols for the controlled switches:



Physical packaging

I don't think there's any industry standard packaging for relays (the way there is for transistors and ICs). Relays seem to use unique cases designed by each manufacturer for each part. Most of these follow roughly the same pattern, though: a rectangular plastic box with a set of pins or terminals on the bottom. Some relays use small round pins or leads suitable for installation in a circuit board, while others use solder terminals for connecting hookup wire.

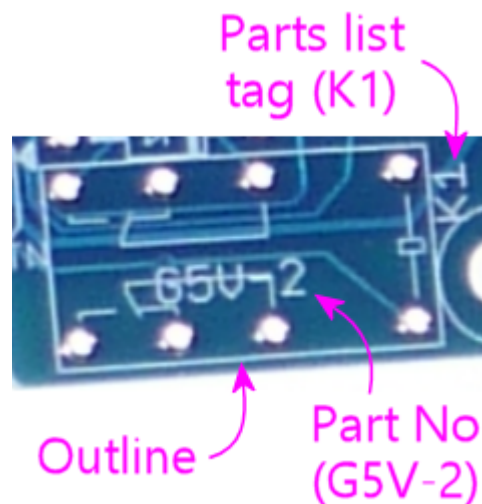


Orientation

All of the relays I've encountered have asymmetrical pin layouts that make it impossible to plug them in wrong. That's true of the ones used in the Pinscape boards. If it fits into the solder pad holes in the circuit board, you've got it right. If it doesn't fit, you've got it backwards, so just rotate it until it fits.

Installing in a circuit board

The Pinscape boards mark the install location of a relay with an outline of the part as seen from above, as usual:



To install, match up the pins on the bottom of the relay with the circuit board solder pads, and fit them into the holes in the pads. The relay should only fit one way (if it doesn't fit, you probably just have it rotated wrong, so try turning it around).

The relay should fit flat against the circuit board when properly seated.

Hold the relay in place so that it doesn't fall out, flip the board over, and solder the pins to the pads on the bottom side.

Selection

When building a circuit board like the Pinscape boards, it's best to stick to the exact part listed. Use the manufacturer part number in the parts list to search for it on Mouser or other electronics vendors.

If you need to find a substitute for a relay listed in a circuit design, you'll need to match several properties from the original relay, which you can find in the relay's data sheet:

- The coil voltage should match exactly. Make sure that it's both the same voltage level and the same type (AC or DC).
- The new relay's coil current should be the same as or lower than the original relay's. You might be able to use a higher current if the circuit board allows it; check with the circuit board designer if you're not sure. In the case of the Pinscape TV ON relay, a coil current up to 600mA is safe.

- The voltage and current limits on the switch should be sufficient for the application. If you're not sure what's needed, it's safest to select a new relay with limits equal to or greater than the original relay's.
- If you're going to install the relay in an existing circuit board design, you'll also have to match the exact package type and pin layout of the original, so that the new relay fits the same solder pad holes in the board. The pin layout for each part should be documented in its data sheet.

77. Transistors

A transistor is a fundamental semiconductor component that acts like an electronic valve, with a control signal that determines how much current can flow through the transistor. This ability can be used for amplifying a small signal into a bigger one and for electronic switching. Virtually all modern electronic computing is based on logic gates constructed from transistors acting as switches.

"Transistor" is an umbrella term for several rather different types of semiconductor devices, based on different physical effects, with considerably different behavior in a circuit. They're all lumped together under the name "transistor" because of the ability they have in common to function as amplifiers and switches.

This chapter is about the **bipolar junction transistor**, which is what people usually mean when they talk about a "transistor" without saying specifically which kind. The BJT was the first type of transistor that was widely produced, and was by far the most widely used until the 1980s, when its dominance was supplanted by a different type called the MOSFET (the subject of the Chapter 78, next chapter). Even though MOSFETs are more numerous these days (mostly because they're the transistors underlying nearly all integrated circuit chips), BJTs are still foundational to modern electronics, and they're extremely useful devices that are still widely used as discrete components (that is, outside of integrated circuit chips). We're including them in this guide for all of those reasons, but more specifically because the Pinscape boards use a few of them.

Static electricity warning

Transistors are sensitive to static electricity. Refer to Chapter 67, Static Electricity Precautions for tips on handling static-sensitive parts.

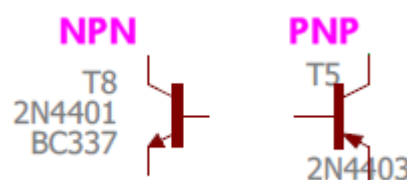
NPN and PNP

Transistors (the BJT type) come in two varieties: NPN and PNP. These are essentially mirror images in terms of the voltage polarities of the signals they work with. The P's and N's in each abbreviation are for "positive" and "negative" (although in a very specific and technical way that you shouldn't take to refer to simple voltage polarities).

The main thing you have to know about NPN vs PNP transistors when you're building a circuit is simply that they're different types, and that you can never swap one for the other. If a particular part in a schematic or circuit board plan calls for an NPN, you must use an NPN transistor there, and likewise for PNP.

On schematics

The schematic symbol for a transistor consists of a thick bar with three lines sticking out, one straight line on one side, and two diagonal lines on the other side. One of the diagonal lines has an arrow, which might point towards or away from the middle bar.

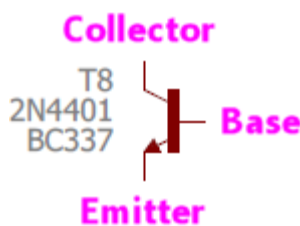




If the little arrow points *away* from the bar, the symbol represents an "NPN" transistor. If the arrow point *towards* the bar, it's a "PNP" transistor.

Note that the little arrow might be shown at top or bottom, and it might be on the left side or the right side. None of that makes any difference - the symbol means the same thing no matter how it's flipper or rotated. Schematic writers will flip the symbol top-to-bottom, or left-to-right, or rotate it at different angles, according to what's convenient to make the lines between nearby connections shorter. It doesn't change the meaning.

The three lines represent the three connections to the transistor, called the base, collector, and emitter:



- The straight line by itself on one side is always the **base** or **B**
- The diagonal line with the arrow is always the **emitter** or **E**
- The other diagonal line is always the **collector** or **C**

Pay attention to where the arrow is, because that's the real key to decoding the symbol. Remember that the symbol can be flipped top-to-bottom or left-to-right, or it can be rotated. But *the arrow is always the emitter*, no matter where it's positioned. The line on the opposite side of the bar from the arrow is always the base, and the arrow-less line on the same side of the bar is always the collector.

Transistors have parts list tags just like other components. These most commonly start with "T" or "Q". As with the "R" tags for resistors and "C" tags for capacitors, these are just arbitrary tags to look up in the parts list, with no other meaning.

Transistors are also usually labeled with the semiconductor ID, like a diode is. In the case of a transistor, this usually starts with "2N". You might also see other part numbers, such as the "BC337" in the examples above. When two numbers are listed for one part like this, it indicates *alternative* parts that you can use - so in the case of T8 above, you could use a 2N4401 or BC337 interchangeably.

On some schematics, the transistor symbol will be enclosed in a circle:



The circle doesn't change anything; it's just an alternative way of drawing the symbol.

Physical packaging

Transistors come in many shapes and sizes. There seem to be about 20 industry-standard package types for through-hole transistors (the kind with leads that you insert through holes in a circuit board). There are probably quite a few more non-standard proprietary packages as well.

The Pinscape boards only use one package type, known as TO-92, which looks like a little black half-cylinder about 5mm on a side, with three leads sticking out the bottom.



Orientation

The TO-92 case is an industry-standard shape and size, but oddly, they didn't standardize the order of the leads while they were at it. If you want to figure out which lead is the base, which is the collector, and which is the emitter, you have to look at the data sheet for the part you're using. For example, if you look at the data sheet for the 2N4401 transistor, it'll include a little diagram like this:

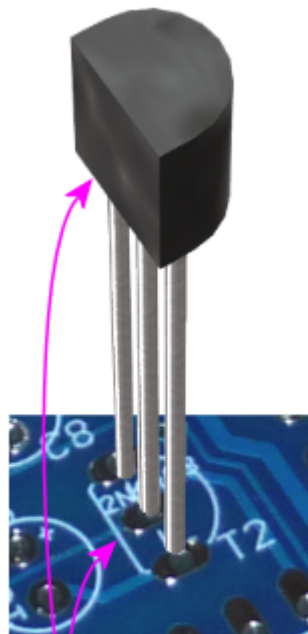
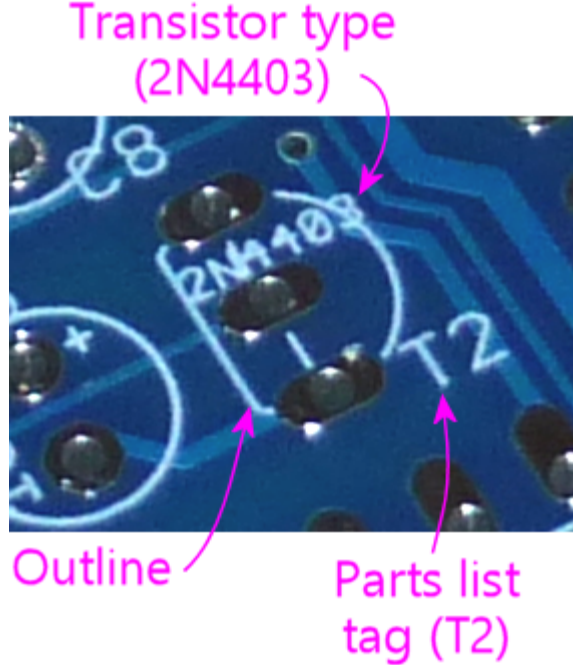


In the diagram, E = emitter, B = base, C = collector.

Other transistors in the exact same TO-92 package might have the leads in a completely different order.

Even though they didn't standardize the lead order for the TO-92 package itself, they *do* always use the same lead order for a given transistor. Every 2N4401 in a TO-92 case will use the same lead order shown above.

Printed circuit boards take advantage of that fixed lead order for each specific transistor type to give you a handy orientation key for each transistor, printed right on the circuit board. If you look at the outline for any transistor in a TO-92 case, you'll see that the outline (as usual) matches the shape of the transistor when you view it from straight overhead. The outline has that same half-circle shape with one side flattened. That flat side is the orientation key. Just line up the flat side on the transistor with the flat side on the printed outline.



Installing in a circuit board

Orient the transistor to match outline printed on the circuit board as shown above. Fit the three leads through the corresponding three holes in the circuit board. Feed the leads through the board until the part is seated as close to the board as you can comfortably get it without forcing it.

Hold the part in place, flip the board over, and solder the three leads to the solder pads. When the solder cools, snip the excess length from the leads.

Selection

Transistors don't have a simple "unit" that describes them, the way we have Ohms for resistors or Farads for capacitors. Instead, schematics and parts lists will specify

a particular transistor to use, by part number.

Transistor part numbers starting with "2N" refer to common, generic transistor types that are made by many manufacturers to the same specs. Many other transistors are identified by a manufacturer's proprietary part number. Manufacturer-specific part numbers don't follow any particular standard format, as they're up to each company to define.

In either case, use the part number listed in the schematic or parts list to search for a matching part at Mouser or another vendor.

It's always best to use the exact part listed, but many transistors have mutually compatible substitutes available. Try a Google search for a term like "2N4401 equivalent" if you can't find the exact original part specified.

All of the NPN and PNP transistors used in the Pinscape boards are used for their switching function. That makes them largely interchangeable with other transistors that are described as "small signal transistors". If you can't find the exact option for a transistor in the Pinscape parts list, you can probably substitute any other "small signal transistor" that meets these requirements:

- It's the same basic type (NPN or PNP) as the original
- Its maximum collector current (I_C , typically listed in the "Absolute Maximums" section of the data sheet) is at least as high as the value listed for the original part in its data sheet
- Its maximum emitter-base voltage (V_{EBO}) is 12V or higher
- Its maximum collector-emitter voltage (V_{CEO}) is 12V or higher
- It has the same case type (TO-92, TO-220, etc), to ensure that the leads will fit in the same solder pad holes in the circuit board
- Its leads (emitter, base, collector) are in the same order, so that you can plug it in the same way. (Alternatively, it can be in the *reverse* order, as long as you remember to rotate it 180° from the way it's depicted on the circuit board when installing it, to match the reversed lead order.)

Note that those rules are specifically for the Pinscape boards. If you're trying to make similar substitutions for other circuit boards, you should those specs for V_{EBO} and V_{CEO} from "12V or higher" to "at least as high as the value listed on the original part list". I was just trying to save you the trouble of looking those up for the Pinscape parts, since in those cases you wouldn't need specs higher than 12V.

Darlington transistors

A Darlington transistor isn't really a different type of transistor; it's just a different kind of physical packaging. But it's worth mentioning because it looks a little different on schematics.

A Darlington is a pair of NPN or PNP transistors, linked together inside a single physical package. To the "outside world", it looks and acts very much like a single NPN or PNP transistor. What makes these devices useful is that the linkage of the two transistors greatly increases the amplification power - the first transistor amplifies the input signal, and the second transistor amplifies that *amplified* signal, so it's like multiplying the two together. A circuit designer can accomplish the exact same thing by wiring two transistors together the same way, but this is such a common trope in circuit design that it's convenient to have it available as a single part. It's one less discrete part to install when you're building a board.

On a schematic, a Darlington is drawn as a pair of regular transistors:



This schematic symbol is so similar to the symbols for two individual transistors that it's kind of hard to distinguish whether it's a single Darlington or two regular transistors. The tell-tale is that there's only one reference designator and part number shown for the pair. The other way you can tell (although less definitively) is that the two individual transistor symbols are drawn so closely together, with almost no "base" line in the second transistor symbol. If they were in fact meant to be discrete parts, they'd probably be spaced out a little more.

Physically, a Darlington is just like a regular NPN or PNP. Like the regular kind, it has three leads, labeled Base, Collector, and Emitter. You install it in a circuit board just like the regular kind of transistor.

78. MOSFETs

A MOSFET is a type of transistor (that's what the "T" in MOSFET is for). MOSFETs are very different from the older kind of transistor we saw in the last chapter (the bipolar junction transistor or BJT), but they're still called "transistors" because they have the same abilities to amplify and switch signals that all transistors have.

Up until around the 1980s, nearly all transistors used in practical applications were BJTs. That completely changed in the 80s, because the manufacturing processes used to create integrated circuit chips favored MOSFETs. MOSFETs are much more prevalent today as a result. They're also widely used as discrete transistors because of their excellent power efficiency. The Pinscape boards use MOSFETs for the high-power switching needed to control feedback devices, which is an application that MOSFETs are particularly good at.

For more on BJTs and transistors in general, see Chapter 77, Transistors.

Static electricity warning

MOSFETs are sensitive to static electricity. Refer to Chapter 67, Static Electricity Precautions for tips on handling static-sensitive parts.

Types of MOSFETs

MOSFETs come in four varieties, combinations of two "modes" and two "channel" types:

- Depletion Mode or Enhancement Mode
- N-channel or P-channel

The mode refers to how the switching function works in the device; depletion mode means that the device conducts when *no* control voltage is applied, and enhancement mode means that it conducts when a control voltage is applied. The difference between N-channel and P-channel MOSFETs is similar to the distinction between NPN and PNP bipolar junction transistors; it refers to the charge polarity of the material used in the semiconducting region.

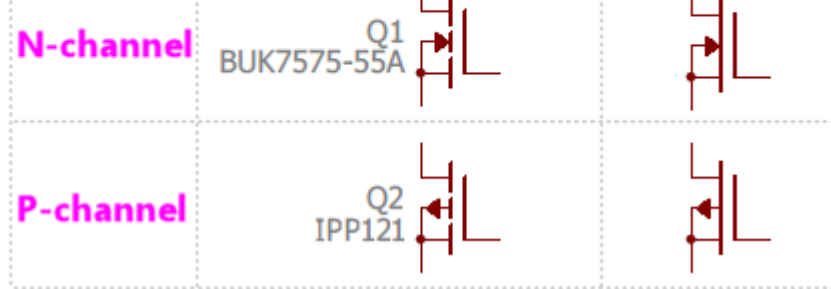
As with NPN-vs-PNP for BJTs, you don't need to know anything about how these variations work to build a circuit that uses MOSFETs. What's important is simply to know that you always have to use the exact mode and channel type called for in the circuit plan. If the plan calls for an "N-channel enhancement mode" MOSFET, you have to use exactly that type.

All of the MOSFETs used in the Pinscape boards are N-channel enhancement mode devices.

On schematics

The schematic symbol for a MOSFET is vaguely similar to the symbol for a bipolar junction transistor, but different enough that you wouldn't confuse them with each other:

	Enhancement Mode	Depletion Mode

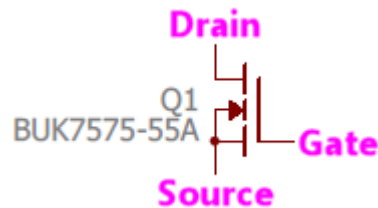


The direction of the little arrow in the symbol indicates whether it's a P-channel or N-channel device. An N-channel device has the arrow pointing inwards, towards the middle bars; a P-channel device shows the arrow pointing outwards.

The mode (enhancement or depletion) is depicted by the main bar: a broken bar indicates enhancement mode, and a solid bar indicates depletion mode.

The N-Channel Enhancement Mode version is the only one you'll see in current Pinscape schematics, but you might see any of the other types in other schematics.

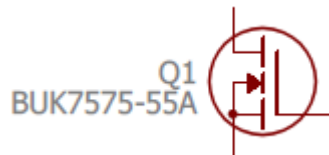
A MOSFET has three connections to the outside world, called the gate, source, and drain.



- The line off by itself on one side is always the Gate
- The line that connects to the little arrow is always the Source
- The remaining line is always the Drain

Note that the gate, source, and drain are often designated G, S, and D on diagrams. If you look at a MOSFET's data sheet to find the order of the case leads, they'll probably label them G, S, D in the diagram.

You might sometimes see the MOSFET symbol enclosed in a circle. That's just a visual affectation; it doesn't change the meaning at all.



Physical packaging

Like any transistor, MOSFETs come in numerous package types. The Pinscape boards only use one package type, though, known as TO-220, which looks like a little black plastic brick with three legs coming out the bottom, and a big metal heat sink fin on the back.

Front view:





Back view:



Most devices in this type of case will be printed with text across the front indicating the part number.

Orientation

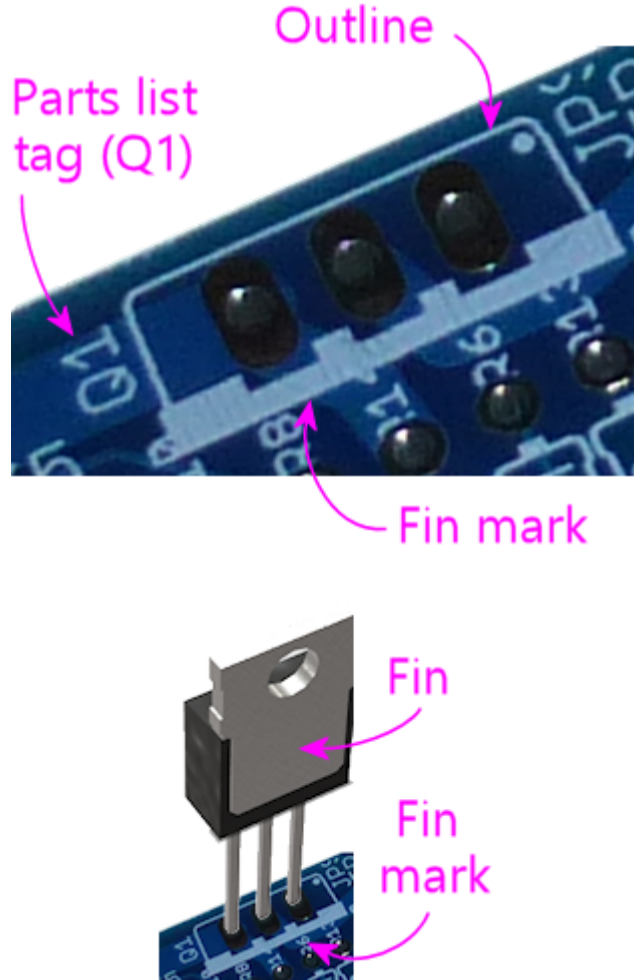
The order of the leads isn't standardized for the TO-220 case, so the only way to tell the order for a particular part is to look it up in the part's data sheet. The data sheet will usually include a diagram like this:



G, D, S stand for Gate, Drain, and Source.

The arrangement of TO-220 leads will always be the same for any given MOSFET, so the Pinscape boards can take advantage of this to show you the correct orientation without forcing you to track down a data sheet. This means you can figure out the correct orientation for a MOSFET just by looking at the circuit board markings

As usual, the Pinscape boards show a little outline of each MOSFET, as viewed from above, where it's meant to be installed. For a TO-220 package, the outline shows a rectangle the size of the plastic brick part, with a heavy line on one side. The heavy line represents the metal heat sink fin. So to orient the MOSFET properly, you just have to line up the fin with the heavy line printed on the board.



Installing in a circuit board

Orient the MOSFET to match the outline printed on the board as shown above. Fit the three leads through the corresponding three holes in the circuit board. Feed the leads through the board as far as you comfortably can.

TO-220 packages usually have little barbs on the leads near the plastic case. This is intentional, to force a little distance between the case and the board to allow air flow for cooling. Just feed the leads through until you reach the barbs, then stop.

Hold the part in place and flip the board over to solder it. Try to keep it at right angles to the board. With the barbs in the leads, it's easy for the part to want to tilt a bit in one direction or the other, so try to keep it straight up and down while soldering.

Solder the three leads to the solder pads. When the solder cools, snip the excess length from the leads.

Selection

Schematics and parts lists will always specify which MOSFET to use by manufacturer part number. There aren't any generic "units" to describe a MOSFET like there are with resistors and capacitors. Use the manufacturer part number to search for a matching part at Mouser or another electronics vendor.

It's easiest to use the exact part specified in the parts list, but as with BJTs, many MOSFETs have compatible replacement parts that can be safely substituted. Try a Google search term like "IPP121 replacement".

The large MOSFETs in the Pinscape boards are used as on/off switches, which makes it fairly easy to find compatible replacements. The parts list, in fact, lists several options for each one. If you can't find any of those and need to look for other alternatives, here are some tips on what to look for:

- The same type (N-channel enhancement mode)
- The same case type (TO-220)
- The same lead order (G-D-S when viewed from the front), so that the part can be plugged into the circuit board slot in the same orientation marked on the board, *or* the reverse lead order, so that you can plug it in rotated 180° from the marked orientation
- Maximum drain current (I_D , normally listed under "absolute maximums" in a data sheet) of at least 6A, or however much current you need the circuit in question to carry
- Drain-source breakdown voltage (V_{DS}) at least 55V, or however much voltage you want to be able to switch on the circuit in question
- Gate-source breakdown voltage (V_{GS}) at least 15V

Even with those criteria in hand, you might find it challenging to search for parts, in part because MOSFET data sheets are packed with a lot of other information, but more because there are just so many MOSFETs on the market. A quick Mouser search shows 20,116 distinct catalog results for "MOSFET"! Even filtering to in-stock, N-channel, enhancement-mode, TO-220 case, $V_{DS} \geq 55V$, $I_D \geq 6A$, $V_{GS} \geq 15V$ turns up over 650 matches. If you don't have working knowledge of MOSFETs beyond the scant introduction we've provided here, you might want to ask someone who does to sanity-check the data sheet for you before committing to a selection.

79. IC Chips

Integrated circuit (IC) chips are ubiquitous in modern electronics, and you'll find several of them on the Pinscape boards. An IC is essentially a miniature circuit board consisting of a collection of the more basic components (resistors, capacitors, and transistors) that we use elsewhere in our own circuit boards. There are ICs for thousands of different functions and applications, ranging from simple things like resistor arrays to complex computing tasks like running Windows.

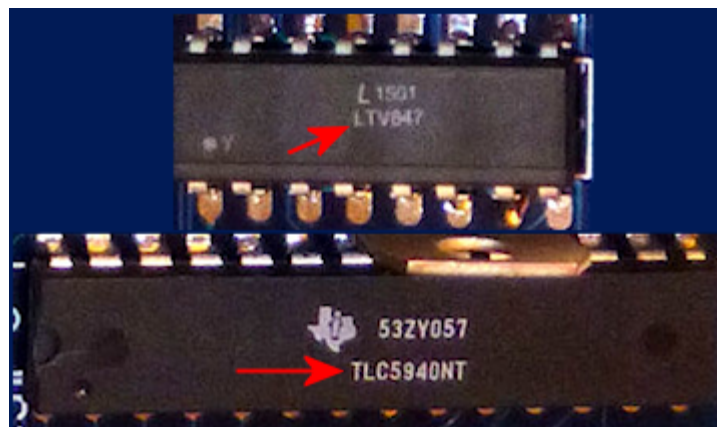
We'll start with some general information that's common to most IC chips, then look at each of the specific chip types used in the Pinscape boards. If you're working on building the boards and you're trying to figure out how to install a particular chip, look for the section on the chip in question later in the chapter.

Static electricity warning

Almost all IC chips are sensitive to static electricity. Refer to Chapter 67, Static Electricity Precautions for tips on handling static-sensitive parts.

Identifying a chip

Most chips can be identified pretty easily by the number printed on the top of the case.



Each chip pictured above has two lines of random-looking alphanumeric strings. This is pretty typical, but it's not any kind of standard; some chips might have more or less printing, which might be arranged in other formats. Whatever the format, the chip name should always be in there somewhere. That should match the name that's used in the schematic and printed on the board. Anything else is usually opaque manufacturing codes not meant for our eyes, such as lot numbers and date codes. How do you pick out the chip name from the other stuff? You pretty much have to figure it from context, by looking for a chip name you recognize. Somewhere in there, you should find the name of the chip as shown in the schematic and on the PCB silkscreen.

Note that the name printed on the chip sometimes has slight variations from the name used in the schematics. In particular, there might be some extra prefix or suffix characters. These usually denote variations of the chip, or different manufacturers, which have already been accounted for in the parts list. Even so, if you do find such a discrepancy, it might be worth double-checking the part list to make sure you haven't accidentally swapped two parts that have similar names.

Chip orientation

Orienting a chip properly before installing it is critical. In the best case, a chip that's installed in the wrong orientation simply won't work, and in the worst case, the error might destroy the backwards chip, or even other components, when you turn on the power. So always make sure you have the right orientation before soldering anything.

Most of the chips used in the Pinscape boards are "DIP" (dual in-line package) chips with two rows of pins sticking out. It's easy to tell which side is the top and which is the bottom: the pointy ends of the pin point down, since they go through matching holes in the circuit board.

What's not as obvious is the proper rotation. DIP chips are rotationally symmetrical: they'll fit the space on the board if you rotate them 180° from their proper orientation. So you have to be careful to get that right.

When you look at the Pinscape boards, you'll see little outlines for all of the chips printed on the top of the board. Most of the IC chip outlines look roughly like this:

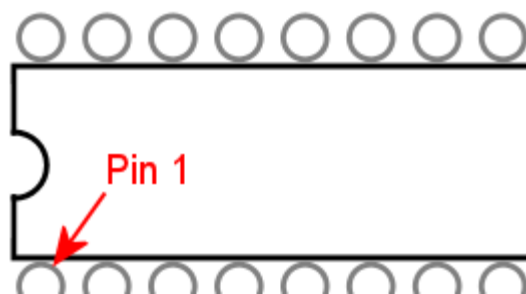


The first thing to note is the printing inside the outline, in this case "PC847". That's the name of the chip to install here. When you're about to plop a chip into position, it's always a good idea to double-check that the name printed on the chip matches the name printed on the board. Different chips can sometimes share the same footprint, so it's good to make the extra check.

Don't be distracted by the fact that the writing is upside-down. The text printed on the board is almost all oriented the same way with respect to the overall board, for the readability's sake. If the text were printed at all sorts of different angles across the board, it could make some of the legends hard to read or even ambiguous (e.g., is that an "N6S" or an "S9N"?). The trade-off is that this often makes the text appear upside-down or sideways with respect to the component it refers to, as in this case. The important thing to remember is to always pay attention to the outline, not the text, to determine which way a component goes.

Which brings us to the second big feature in the photo: the notch. That's the most important feature of the outline, because it's your key to orienting the chip properly.

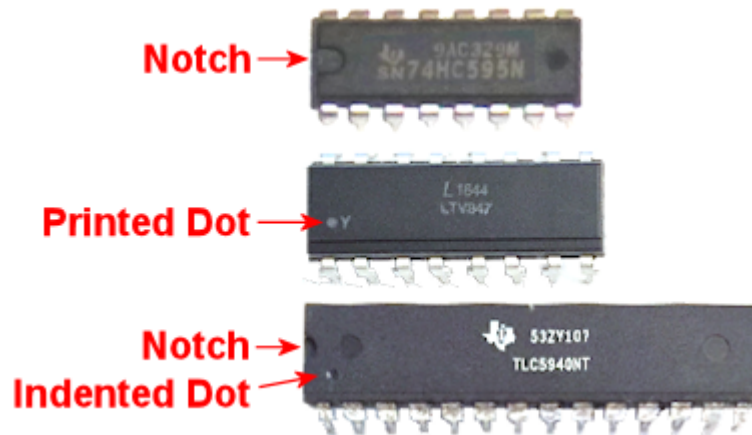
Whenever you see a notch in a chip outline like this, turn the board so that the notch is at the left like in the photo. In that orientation, pin 1 is the bottom left pin, just below the notch.



Now you know where pin 1 goes on the board, so all you have to do is match that to pin 1 on the chip.

Some chips have the same type of half-circle notch that's printed on the circuit boards. The notch in the chip isn't usually printed in ink, though - it's usually a slight indentation or depression in the plastic case. When you find a notch on the chip, orienting it to the diagram on the board is easy: just line up the notch on the chip with the notch printed on the board.

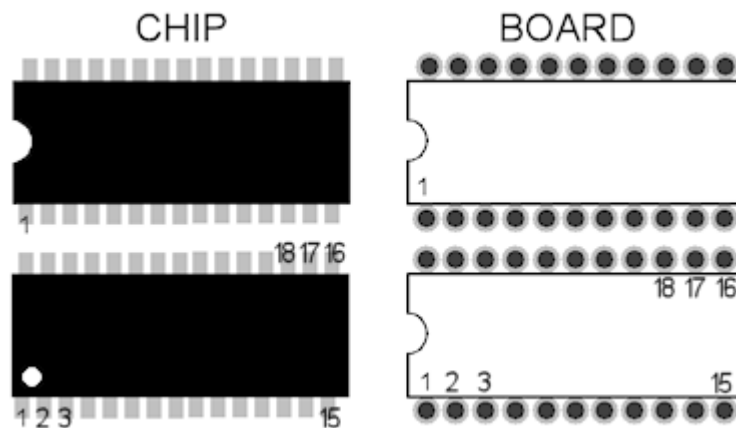
Not all chips use the notch, though. The other common convention is a little circular "dot" in one corner. The dot is sometimes marked in ink, and other times it's just a subtle indentation in the chip casing. When there's a dot, it's in the corner nearest to pin 1.



Common orientation markers on IC cases. Some ICs have indented notches on one side; simply line this up with the notch in the printed outline on the circuit board. Other chips use a "dot" at one corner of the chip to mark the location of pin 1. The dot might be printed in white ink, or might be indented in the plastic.

If you don't see the notch or dot, it might just be a really subtle one that's hard to spot. Look at the chip under a strong light, and hold it at different angles. The indentation for the notch or dot is sometimes very slight.

Once you find the notch or dot, it's just a matter of lining up pin 1 on the chip with pin 1 on the board.



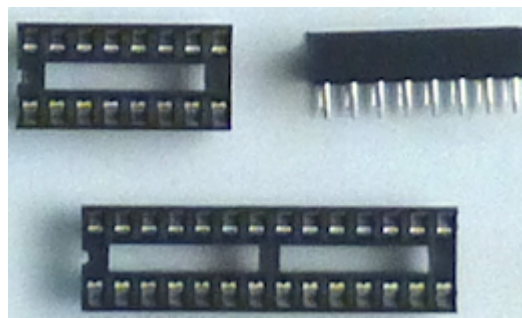
Examples of how to match the chip marking with the board marking to find the

proper orientation. Top: the chip has a notch in its case like the one printed on the board, so simply line up the notch on the chip with the one on the board. Bottom: The chip has a dot near one corner. Orient the chip so that the dot lines up under the notch printed on the board.

Sockets

There are two main ways to install a DIP-type chip on the circuit board: solder it directly to the board, or use a socket.

A socket is a receptacle that matches the exact footprint of an IC chip. It has pins coming out the bottom, matching the pins on the chip. You fit the socket's pins through the holes in the board intended for the chip, and solder the socket pins to the board. The top of the socket has matching openings for the chip's pins. Once the socket is soldered to the board, you just plug the chip into the socket. The pin openings in the socket are spring-loaded, so they hold the chip in place without having to solder the chip.



IC sockets: top view of a 16-pin socket; side view of 16-pin socket; top view of 28-pin socket. Note the notch at the left side in the top views: this has the same purpose as the notch printed on the circuit board outline of the chip, to serve as an orientation guide. Line up the notch on the socket with the notch printed on the board when installing the sockets.

The main advantage of using a socket is that it makes it practically effortless to remove and replace a chip. That's a great time-saver if you find that one of your chips is defective, or if it ever fails later, or if you accidentally install it backwards on the first try. In contrast, it's extremely difficult to remove a chip that's been soldered directly to the board. It only takes a tiny bit of solder on each pin to make it stick, and it's next to impossible to get all of the solder off all of the pins on a large chip.

I recommend using sockets for all of the large chips, but it does slightly increase the cost, so some people prefer to solder the chips directly.

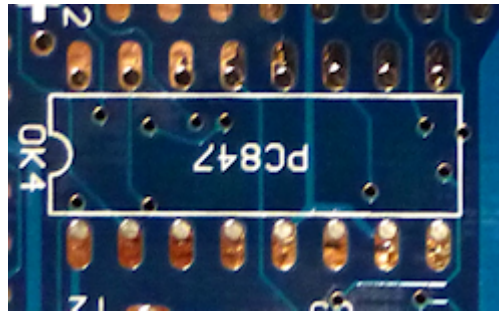
If you use sockets, note that each socket has a notch cut into the plastic. This is meant to mimic the notch in the chip outline printed on the circuit board, to serve as the orientation marker. You should be sure to line up the notch in the socket with the notch printed on the circuit board when you install each socket. That way, you can use the notch in the socket as a proxy for the notch in the circuit board chip outline when plugging in the chip - which is important because the printed outline will probably be hidden behind the socket once the socket is in place.

How to install a chip on a circuit board

If you're using a socket, follow the procedure we're about to describe for soldering the chip, and just substitute the socket for the chip in all of the steps. After the

socket is installed, plug the chip into the socket, taking care to orient the chip properly as described above.

To solder a chip (or socket) to the board, first find the outline printed on the board for the chip. Most IC chip outlines look roughly like this:



Check that the chip type printed on the board matches the chip you're getting ready to install.

Line up the chip's or socket's pins with the holes along the edges of the outline. The number of holes should match the number of pins. Carefully insert the pins through the holes.

Make sure the chip is oriented properly, as described above. This is an excellent time for the "measure twice, cut once" rule - check and double-check that you have the chip turned the right way.

Now insert the pins into the holes.

With DIP chips (the ones with two rows of pins), the pins are usually angled out just a little wider than the holes, so you usually have to bend one row of pins inward very slightly to fit them through the holes. To make this easier, I like to start by inserting one row of pins first. Then you can apply a little pressure to the whole chip to uniformly bend the now-seated pins enough to fit the opposite row through its holes. (Sockets don't usually require this kind of maneuvering, since their pins stick straight down. It's another way sockets make things a little easier.)

Inspect the pins from the top to make sure they all got seated properly. It's easy for one or two pins to miss their holes and go sideways when you seat the rest. If this happens, you might be able to nudge the missing pins into their holes if they're not too far askew, but don't force anything. The pins are delicate and don't stand up to much bending and re-bending. If necessary, take the chip back out, carefully (very carefully) straighten any pins that went sideways, and try again.

Once you're satisfied that all of the pins made it into their respective holes in the board, hold the pin in place from the top and flip the board over. Verify that all of the pins really made it through the openings, as seen from the bottom of the board.

If everything looks good, it's time to solder the pins in place. Keep the board flipped over and solder the pins from the bottom.

Solder a pin at one corner first (any corner will do). You should hold the chip firmly in place against the board from the other side during this step to make sure that gravity isn't pulling it a little away from the board.

Check *again* that the pins are still all in place. They can sometimes work loose during all of this board flipping and soldering.

If everything is still in place, solder the pin at the diagonally opposite corner next,

still holding the chip pressed firmly against the board from the other side.

Do one more check that the pins are all still where they should be. This is basically the point of no return - it's not too difficult to get the chip free if necessary with only the two pins soldered, but it'll be practically impossible once you solder more pins. So it's worth making sure that everything is good before going on.

At this point, the two attached pins at the diagonal corners should be enough to secure the chip mechanically, so you shouldn't have to worry about anything coming loose from this point forward. You can just work through the rest of the pins one at a time to solder each one in place.

Pin numbering

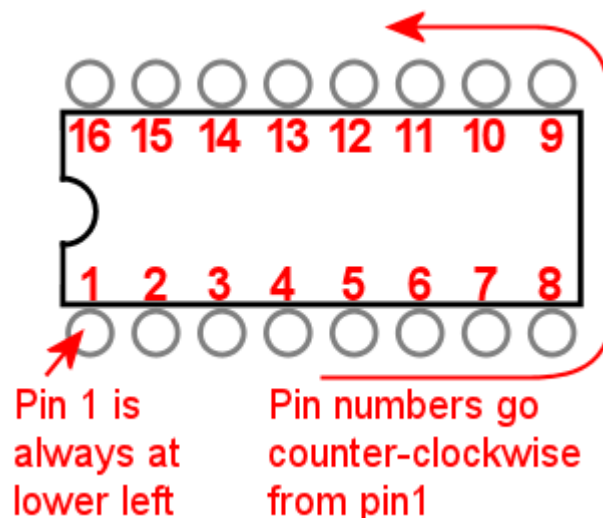
All of the pins on a chip are numbered, for the purposes of identifying them in the schematic.

The pin numbers are irrelevant when you're installing a chip, since all you care about is getting the orientation right. However, you might find a need to cross-reference the individual pin connections in the schematic with the physical boards if you ever have a problem that requires debugging with a voltmeter. If you ever have a problem with the boards, one of the first debugging tasks will likely to be check the continuity between various points on the board, to make sure that pins that ought to be connected actually are connected.

Fortunately, pin numbering on the physical chips is pretty straightforward, and better yet, it's highly consistent across different kinds of chips.

The Pinscape boards mostly use DIP chips - the type with two rows of pins on opposing sides of the plastic case. All DIPs follow these rules:

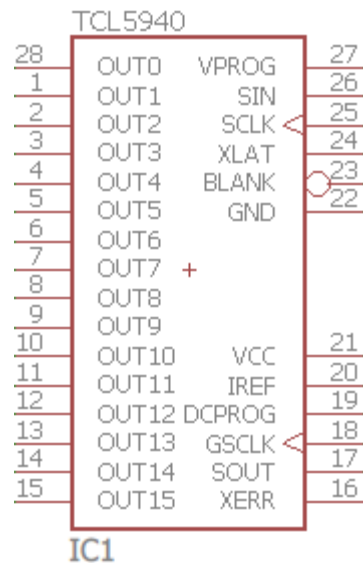
- Pin 1 is at the lower left (with the chip properly oriented)
- Pin numbers increase **counter-clockwise** around the chip



Chip symbols on schematics

The schematic symbols for chips can vary a little bit. There are a few types of chips that have special symbols because of the function they perform. Most chips, however, are so specialized that there's no special symbol for them, so they're shown on schematics quite generically, as simple rectangles with lines sticking out to represent the pin connections. For example, here's the symbol for a TLC5940, which

is the large PWM controller chip used on the Pinscape expansion boards for feedback device outputs:



The rectangle represents the TLC5940 package, and the little lines sticking out from the sides represent the pins. Note how every line has a number. Those are the pin numbers, and they correspond to the physical pin numbers we described above. You can use those numbers to match up every pin on the schematic with the corresponding physical pin on the board, which is important when you're trying to debug a problem.

If you look closely, though, you'll see that the numbers shown on the schematic aren't in the same order as the physical pins. That's in keeping with the whole idea of a "schematic" - an abstract representation that only keeps the essential information. Even so, it might seem like it would be simpler if they'd use the same pin ordering, but there's a reason they don't. To understand the reasoning, look at the labels inside the TLC5940 box. Those are the "names" of the pins, which are just arbitrary mnemonics that are there to help a circuit designer remember the function of each pin without having to memorize all the numbers or constantly refer to the data sheet. Notice how the whole left side is OUT0, OUT1, OUT2, etc. Those pins are all of the PWM outputs. On the physical chip, they're not all together, but they're grouped on the schematic. That keeps the schematic drawing a little neater.

In any case, the important thing to take away from this is that you shouldn't pay any particular attention to the order of the pins shown on the schematic; just pay attention to the numbers. Every pin's number is explicitly shown, so you don't have to remember a counter-clockwise or anything else; you just look at the numbers printed there.

In the sections below on the specific chips making up the Pinscape boards, we'll show each chip's exact symbol so that you can more easily recognize it on the schematics.

Selecting chips

When you're ordering parts, the basic rule for IC chips is that you should exactly match the chip name shown on the parts list.

This doesn't mean you have to get the exact Mouser part number listed. That's different from the chip name; the Mouser part number is Mouser's catalog ID, which encodes the manufacturer as well as the chip type. Many chips are only made by a

single manufacturer, so in those cases they amount to the same thing. However, some chips are generic, and interchangeable versions are made by several different manufacturers. In these cases, the different manufacturers will all use the same chip name, because that describes the specific function and electrical characteristics of the chip, but the different version will have different Mouser catalog numbers. So the thing to pay attention to is the chip name.

If you find parts that have similar but not identical chip names, it's better to err on the side of caution and assume they're different. There are some very different chips with confusingly similar names out there. If you think you found a match with a slightly different name, the only way to be sure is to carefully compare the data sheets for the two parts and make sure they really are functionally equivalent.

Chips on the Pinscape boards

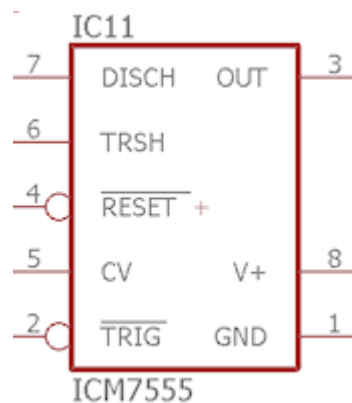
555/7555 timer

The 555 is a venerable and widely-used timer chip. The Pinscape boards use it (or more specifically, a variant called the 7555) to implement the "timer-protected outputs" for the replay knocker and chime outputs.

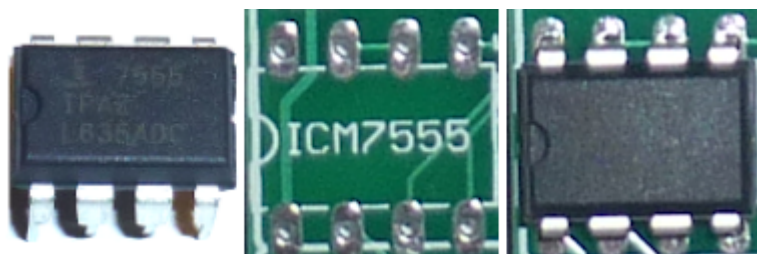
The 7555 is a more modern CMOS version of the original 555. The Pinscape boards use the 7555 because it integrates better with logic circuitry than the regular 555.

When buying parts, be sure to buy the 7555 when the parts list calls for it, not the original 555 or other variant. The variants all come in the same package and have the same pin layout, so they'll physically fit the sockets, but there are some differences in their electronic properties.

On a schematic, a 7555 is depicted with the generic IC box diagram, with eight pins. "ICM7555" is printed near the box to identify the chip type. The component name for a 7555 is of the form IC*n*.



The physical chip is an 8-pin DIP. My samples have an easily visible notch for orientation. To install on the circuit board, just line up the notch on the chip with the notch in the chip outline printed on the circuit board.



7555: chip package, circuit board outline, and chip installed in circuit board. Line up the notch on the chip with the notch printed on the circuit board outline to orient the chip properly. The writing on my sample chip is so faint that you can barely see it in the left photo, and can't see it at all in the right photo, but you can see it on the actual chip with the right lighting.

LD1117AV33 3.3V regulator

The Pinscape boards use a type of chip known as a voltage regulator to supply 3.3V to some of the logic chips on the boards. The part name for the 3.3V regulator we use is LD1117AV33. Similar regulators are available for numerous other voltages, but the Pinscape boards currently only use the one type.

When buying parts, don't try to "fuzzy match" the name of this chip with similar-looking parts, because suffix in this case ("AV33") is highly significant: it indicates the regulated voltage. That's a critical element of the circuit design. Similarly named chips with slightly different suffixes regulate to different voltages, so they won't work as substitutes.

These chips don't come in the usual DIP form factor. Instead, they use a type of package more commonly used for transistors, known as a TO-220. Here's what it looks like:



Front and back view of LD1117. Note that the part name is printed on the front of the plastic case to help identify the part.

Note that the TO-220 package type is widely used for other, completely different components, particular MOSFETs and power transistors. Anything in a TO-220 looks just like this, so you can't identify an LD1117 by the shape of the case alone. For positive ID, check the markings on the case. For this part, the chip name (LD1117AV33) should be printed on the front of the plastic case.

On the schematic, these chips are shown with the standard generic IC box diagram, with three pins. However, unlike most IC box symbols, these don't show any pin numbers. They only show mnemonic labels for the pins. The reason is that there aren't any standard pin numbering conventions for the TO-220 package used for this chip, so pin numbers would only be confusing. This is a case where you have to look at the data sheet to figure out the correspondence between the pins on the schematic and the physical pins on the device. But we'll save you the trouble:



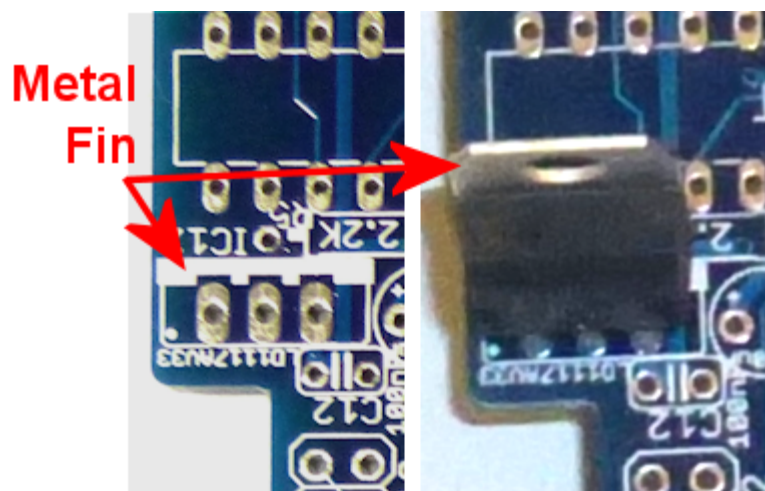
Above left: the LD1117 voltage regulator symbol on a schematic. Above right: diagram of the package showing how the physical pins relate to the pins on the schematic symbol. The package diagram shows the "front" of the package, with the black plastic case facing the viewer. The back of the chip is the big metal fin, visible in the diagram sticking out from the top.

The component ID shown on the schematic for these chips uses the typical form for IC chips, IC n .

On the circuit board, the LD1117 doesn't use the standard notched-rectangle outline, in keeping with its unusual packaging. Instead, it shows an outline with a heavy bar on one side, which represents the big metal fin on the back of the chip package:



To install the chip in the circuit board, orient it so that the metal fin on the back of the chip lines up with the heavy bar printed on the PCB outline. The heavy bar in the outline represents the fin, so you just need to make sure the actual fin is oriented on the side indicated on the outline.



When you install the chip in the board, note that the plastic package won't quite sit flush against the board. The legs have kinks near the tops (the case side) that act as stops, which will keep the plastic case part a couple of millimeters above the board surface. That's perfectly normal; don't try to force the kinks through the holes. The extra distance from the board is there by design, to help the fin radiate heat more

efficiently. You can be sure that any part you see with a big metal fin like this is something that gets hot in normal use, and the fin is there as a heat sink.

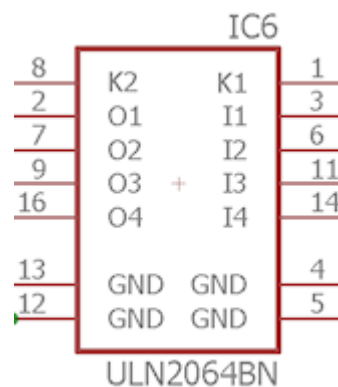


To install this chip, insert it through the holes, checking that the fin is oriented to match the heavy bar in the outline printed on the board. Flip the board over (taking care to hold the chip in place so that it doesn't fall out), and solder the three pins to the pads from the bottom of the board. The leads on these chips are quite long and will stick out about a centimeter from the back of the board when you're done, so you should trim the excess with wire cutters after the chip is in place. Trim the leads to be roughly flush with the top of the solder ball.

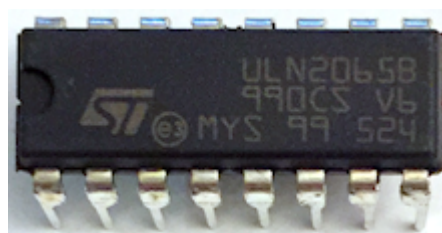
ULN2064B Darlington transistor array

The ULN2064B is an array of four Darlington transistors. Darlington transistors are high-gain transistors that can be used for amplifiers, or in our case, switching medium-power loads from logic circuits. These chips can handle loads up to 1.5A on each output. The Pinscape main boards use these for the flasher LED outputs, because they have plenty of power capacity for large LEDs and are physically compact.

This chip uses the standard generic IC box symbol on schematics, with 16 numbered pins. The component name shown on the schematic uses the form IC*n*.

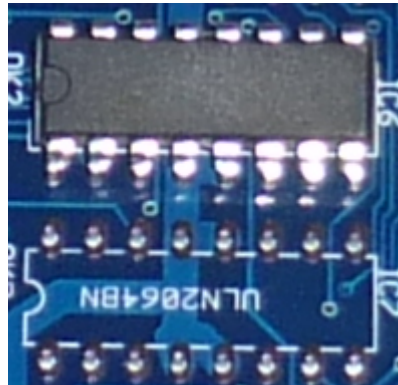


The physical chip is a standard 16-pin DIP.



ULN2065B 16-pin DIP package. The half-circle notch (visible at the left edge of the package) serves as the orientation marker when installing. Line up the notch on the chip with the notch in the chip outline printed on the board. Note that this is the ULN2065B, which can be substituted for ULN2064B.

On the circuit board, the location for this chip is shown with the usual chip outline, with "ULN2064BN" printed in the outline. Line up the notch on the chip with the notch in the printed outline to orient the chip properly.



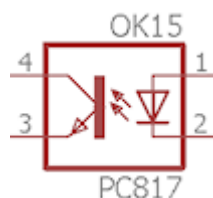
Substitutions: The ULN2065B can be substituted for the ULN2064B. The two are almost identical chips, the only difference being that the 2065 is rated for higher maximum voltage. In other words, the 2065 is just a slightly tougher version of the same chip.

PC817 optocoupler

The PC817 is an optocoupler, which is a device that connects two circuits via light signals rather than electronic signals. The light signals are transmitted by a tiny IR LED inside the chip, and are received by an adjacent phototransistor. (All of the light transmission happens inside the chip, so you won't see light coming out of it, and you don't have to worry about interference from ambient light. An optocoupler isn't the same as an "opto interruptor", which is a kind of switch that's controlled by blocking and unblocking a light beam, like an electric eye. An optocoupler doesn't have an exposed beam that you can block.)

This might sound like a lot of trouble - turning electrons into photons, and then turning the photons back into electronics. But it serves a very useful purpose: it lets the two circuits transfer signals without any electrical contact. That has many applications, but the one we use it for in the Pinscape boards is to create a sort of safety barrier between logic circuits and power circuits, to help protect the logic circuits from the higher voltages and currents used in the power circuits.

On a schematic, the PC817 looks a little different from other ICs, because it doesn't use quite the same generic IC box. Optocouplers are important enough in electrical engineering practice that they have their own special symbol:



If you ignore the interior of the box, you'll see that this actually does still follow the

same pattern as the generic IC box: it's still a box with lines attached representing the IC pins, and the pins are numbered as usual. What's different is that the interior of the box shows symbols instead of mnemonic labels for the pins. To an engineer who knows the language, the symbols are the equivalent to the mnemonics used on other chips, in that they indicate the functions of the pins. The symbol on the left side of the box represents the phototransistor that receives the optical signal; the symbol on the right represents the LED that transmits the signal; and the two diagonal arrows in the middle represent the photons carrying the signal from the one side to the other.

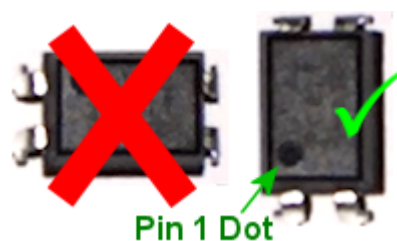
You might also notice that this chip's component ID is "OKn" instead of the usual "ICn". This is an EAGLE-ism; "OK" probably stands for "Opto-Koupler", and I'm not sure why they chose "OK" rather than "OC", but at a guess it's to avoid confusion with "C" for "capacitor". At any rate, just be aware that the schematics and parts lists use this unusual "OKn" designation for these chips, even though they're like any other ICs for all practical purposes.

The physical chip is a 4-pin DIP:



The orientation marker on these chips is usually the "pin 1" dot, in the corner nearest pin #1. The chip in the photo above uses the indentation form of the dot, but yours might have a printed white dot instead. As always, the dot can be such a subtle indentation or faint ink mark that it's tough to see without a strong light and/or magnifying glass.

This chip's proportions are unusual for a DIP, which can be confusing. You get accustomed to all the other DIP chips being wider than they are tall. So it can be tempting to think that this one needs to be rotated into "landscape mode", with the pins at the left and right sides. That's even the way the printing on the chip is aligned, because of the limited space. But consistency is the key here: this is still a DIP, so apply the standard DIP rules. When properly oriented, the pins go along the top and bottom edges, and the pin 1 dot goes at the lower left corner.



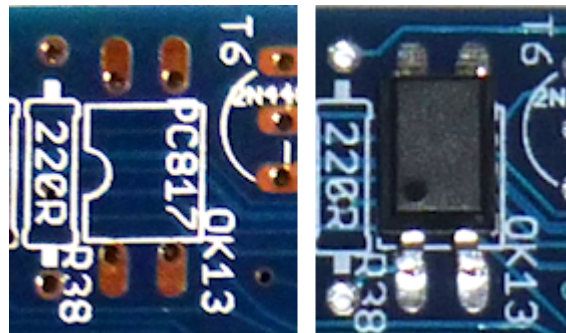
Orienting the PC817. Even though its "aspect ratio" is unusual for a DIP, use the same orientation rules you'd use for any other DIP chip, placing the pins along the

top and bottom edges and the "pin 1" dot at lower left.

On the circuit board, the slot for a PC817 is marked with the standard IC outline, with the notch on the left side and "PC817" printed inside the outline to indicate which part to install.



Install the chip like any other DIP. Hold the board so that the notch in the printed outline is at the left side, and orient the chip with the pin 1 dot at lower left.

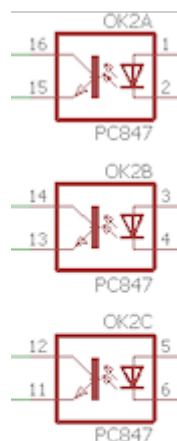


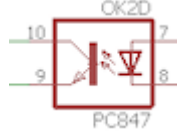
Substitutions: The LTV817 is equivalent to PC817.

PC847 quad optocoupler

The PC847 is a chip that consists of four PC817 units packaged into a single IC. There's really no difference at all electrically or functionally between one PC847 and four PC817 chips, but the quad package is a little more convenient to work with when multiple optocouplers are grouped in the same area of the board, simply because it reduces the the number of parts you have to solder.

On a schematic, a PC847 is even represented just like it's four separate PC817 chips. Which it really is, functionally speaking, and schematics are all about function.





The schematic symbol for a PC847 chip consists of four separate PC817 symbols. The only clue that they're the same physical chip is the component ID: look for the A-B-C-D suffix, which tells you the sub-unit within the chip.

There are a couple of things to notice. The first is that the individual units all have the same "OKn" component ID, but each one also has a suffix - A, B, C, D. The suffix tells you which sub-unit we're talking about. The common OKn prefix is how we know that these units are all part of the same physical PC847 chip. In my own schematics, I also make a point of grouping the sub-units together on the page, so that you don't have to go hunting around to find all of them, but you might encounter cases in other people's schematics where the sub-units aren't grouped.

The second thing to notice is the pin numbering. Note how the pin numbers aren't grouped contiguously on each sub-unit. For example, unit A uses pins 1, 2, 15, and 16. If you scan over all four sub-units, you'll see that all 16 pins are accounted for (and each appears exactly once).

The chip is packaged as a standard 16-pin DIP:

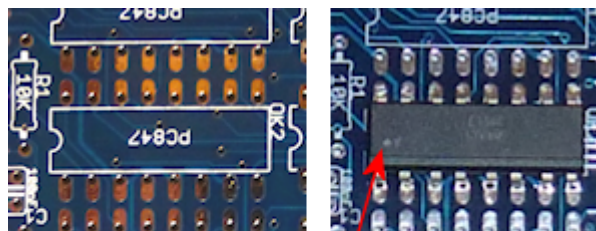


LTV847 (equivalent of PC847). These chips use a (faintly) printed dot to mark the location of pin 1.

On the circuit board, a PC847 is marked with the standard IC chip notched outline, with "PC847" printed inside to identify the part to install there.



Installing a PC847 is just like any other DIP. Orient the board so that the notch in the printed outline of the chip is at the left side, then orient the chip with the pin 1 dot at the lower left corner.





Pin 1 dot

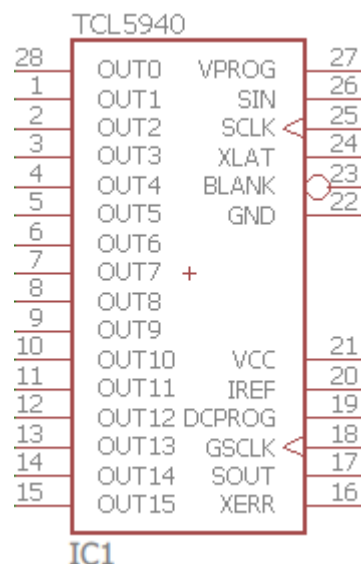
Substitutions: The LTV847 is equivalent to PC847.

TLC5940NT PWM controller

The TLC5940NT is a PWM (pulse-width modulation) controller chip. This means that it generates a series of very fast and precisely timed on/off pulses, under the control of a computer or microcontroller. PWM has many uses, but for our purposes, it's a way to control the brightness of a lighting device, the speed of a motor, or the force of a solenoid. The Pinscape expansion boards use these chips to implement the output ports on the main board and the power board.

The TLC5940 is a proprietary chip from Texas Instruments, and TI no longer manufactures the DIP version used on the Pinscape boards. They do still manufacture the same chip in "surface mount" packages, but those aren't nearly as hobbyist-friendly as DIPs, so the Pinscape boards still use the DIP version for ease of assembly. Fortunately, the DIP version has always been extremely popular with robotics and Arduino hobbyists, and perhaps as a result the supply continues to be plentiful. That continues to surprise me, since TI hasn't made the DIP version since about 2014, but some people conjecture that the continuing supply is coming from gray-market versions manufactured by (presumably unauthorized) third-party factories. Whatever the reason for this good luck, the Pinscape boards will continue to use the DIP version as long as it remains easy to find. If that ever changes, I'll update the boards designs to use one of TI's newer PWM chips instead. That will make assembly a little trickier, because those are all surface-mount parts, but on the plus side, TI's newer PWM chips have much nicer designs than the rather aged TLC5940.

On a schematic, the TLC5940NT appears as the standard generic IC box, with 28 numbered pins.

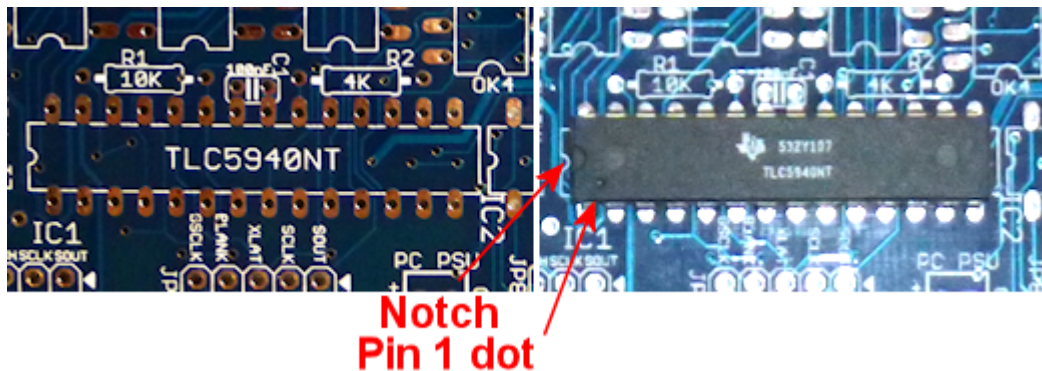


The physical chip is a 28-pin DIP.



Note that the sample pictured above has both the half-circle notch at the left side and a "pin 1" dot, both as slight indentations in the plastic. You might also notice larger, shallower circular depressions centered vertically at either end of the chip. Ignore these; I think those are just artifacts from the molding process that carry no meaning. Just pay attention to the standard notch and dot markers.

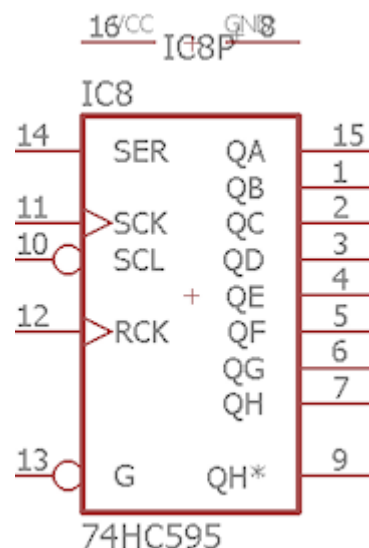
On the circuit board, the TLC5940NT is marked with the standard notched outline for an IC, with "TLC5940NT" printed within. To install the chip, orient the board so that the notch in the printed outline is at the left side, and then orient the chip so that pin 1 is at lower left.



74HC595 shift register

A shift register is a logic chip that lets a microcontroller set the on/off voltage states for a number of pins on the chip. The Pinscape chime boards use these chips to control the outputs on the chime board.

On a schematic, the 74HC595 is drawn with the usual IC box:



There is one unusual feature of this symbol that you might not even notice if you saw it as part of a larger schematic, but it stands out when we isolate the symbol like this. The unusual part is that separate-looking bit at top. Above the chip box, the symbol shows two of the IC's pins that *aren't* included in the box. The way you can tell that those disembodied pins are part of the same chip is the "IC8P" marking. The "P" suffix tells you that these pins constitute a part of "IC8"; the "P" is for "power", as these are the power and ground connections for the chip.

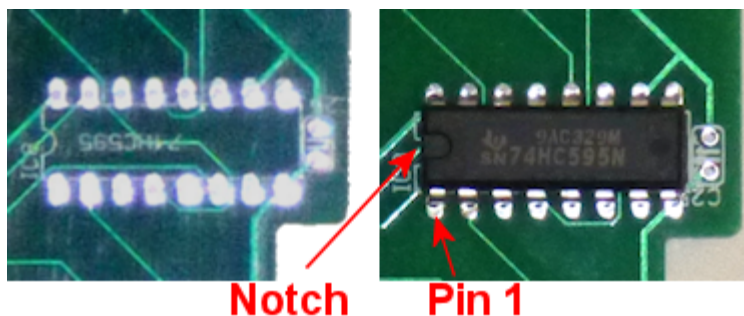
This use of separated schematic elements is somewhat reminiscent of the PC847 that we saw earlier, where the individual optocoupler units within the chip are drawn as separate boxes. In this case, the pins are separated simply to reduce clutter within the box. The symbol's designer thought that there were enough pins already and wanted to get a few out of the way. The power and ground pins were elected because they're givens in any chip. Some engineers just short-hand them away entirely, leaving them off the schematics and letting the reader assume they're connected in the standard way. I prefer showing them explicitly, which in this case that means adding this little islands of pins separated from the chip. I consider this style rather obfuscatory, and fortunately this is the only chip in the Pinscape boards that uses it.

The physical chip is a 16-pin DIP.



My samples have an indented notch on one side of the case to indicate pin 1: as always, if you orient the chip so that the notch is at the left side, pin 1 is the bottom left pin.. As always, this can vary by lot, so yours might have a painted or indented "pin 1 dot" instead.

On the circuit board, this chip uses the standard IC notched outline, with the part name (74HC595) printed within to identify the component to be installed there. To install, orient the board so that the notch in the printed outline is to the left, then orient the chip so that pin 1 is at the lower left.



80. Connectors

One of my "design rules" when I built my own cab was that everything should be modular: pieces should be self-contained, and it should be easy to take anything out for servicing or replacement without major surgery, and just as easy to put it back. In the case of the electronics, this meant that nothing should be "hard-wired" into the cabinet. I should never have to cut a wire or unsolder anything just to remove an electronic device. The way to achieve that is to make sure all of the connections in the wiring are made with modular connectors that can be plugged and unplugged, just like a power plug in a wall outlet.

When I started planning my cab, then, I knew I wanted to use some kind of pluggable connectors throughout, but I didn't have enough electronics experience to know exactly what to buy. I looked to see how other cab builders handled this, and found that there aren't any widely agreed "best practices" in this area. Looking at Mouser to see what's available only muddled the waters further, because they have literally thousands of options for connectors. Worse still, many of the available connectors are themselves fairly complex systems, where you have to assemble plugs and sockets out of multiple. It's hard to know where to begin when faced with the vast array of available options.

That's why I'm devoting a whole chapter to this seemingly trivial subject. In the course of building my own cab, I managed to narrow the huge range of available connector options to a few versatile types that have served me well. The Pinscape parts lists use some of these directly, so some of the information we'll cover is simply to help you recognize and use the parts required for the Pinscape components. We'll also look at some of my go-to connectors for other miscellaneous connections throughout the cabinet, to save you the trouble of repeating all that research.

0.1" pin headers

This is one of the most ubiquitous connectors that you'll see in hobbyist electronics, and really electronics in general. A pin header is basically just a row of connector pins sticking up out of a circuit board, to provide a connection point to the outside world for a group of circuits.

You plug into the pin headers with matching connectors. In most cases, the connector attaches to a ribbon cable or hookup wires that lead off to whatever is meant to be connected to the board. This lets you plug and unplug the whole collection of circuits represented by the header with a single connector.

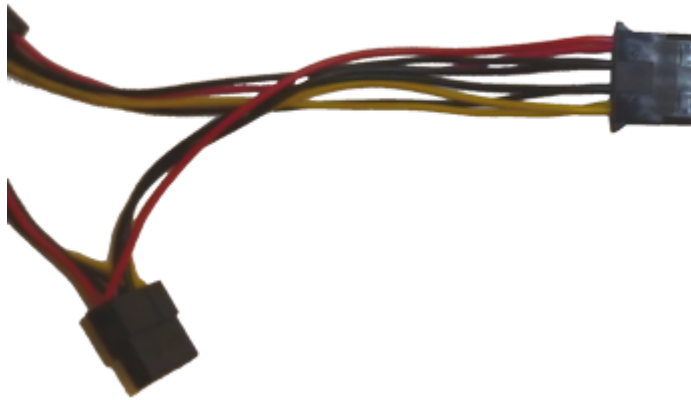


Pin headers can be out in the open, as in the top two examples at right, or can be enclosed in a plastic "shroud", like the bottom example at right. The pin spacing is the same either way; the shroud is there primarily to make sure you can't orient the mating plug the wrong way when plugging it in. The little slot you can see in the picture of the shroud at right lines up with bump on the connector, so if you try to insert the connector backwards, the bump hits the other side of the shroud and prevents you from inserting it.

Pin headers and the matching connectors are described in more detail (with product links and installation instructions) in Chapter 81, 0.1" Pin Headers.

PC disk power connectors

An ATX power supply comes with an array of power cables with different connectors on the ends. If you want to use an ATX supply to power feedback devices or audio equipment, the easiest way to tap into its power outputs is via the disk connectors. These are the four-pin female connectors that look like this:



The four pins are connected to color-coded wires. You can identify the pins by the wire colors:

Wire Color	Voltage
Black	0V (Ground)
Red	+5V
Yellow	+12V

The easiest way to connect to these is to snip the connector off the end of one of the cables, strip the wires, and solder your other wires directly to the ends, or connect them through screw-terminal blocks. But that's not great because it permanently modifies the power supply.

A better way is build your own mating connectors and just plug into them, the way you'd plug in a PC disk. It's pretty easy to build the matching connectors. Here are the parts you need:

- TE/AMP 1-480426-0 4-pin housing
- TE/AMP 60620-1 male crimp pins, quantity 4 per housing

Those are crimp-pin housings, so see Chapter 82, Crimp Pins for instructions on how to assemble them.

Also see Chapter 45, Power Supplies for Feedback for more on using ATX power supplies for feedback device power.

PC motherboard "front panel" connectors

Most modern PC motherboards feature a standardized connector for the miscellaneous case buttons and indicator lights - the soft on/off button, reset button, power light, disk activity light, etc. This is known as the F_PANEL or Front Panel connector, and is usually labeled as such on the motherboard.

Intel defined a standard layout for this connector, so that motherboards and cases

can easily inter-operate without proprietary connectors. It happens to use the standard 0.1" pin header layout, with a 10-pin (dual-row, 2x5 pins) header.

You can plug into these with one of the 0.1" crimp pin housings described above.

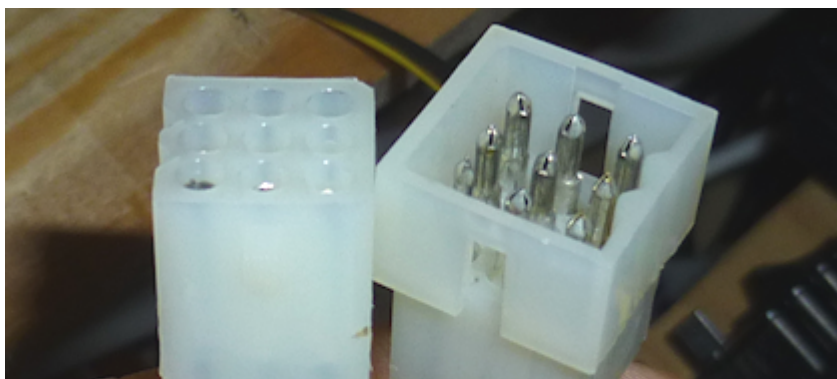
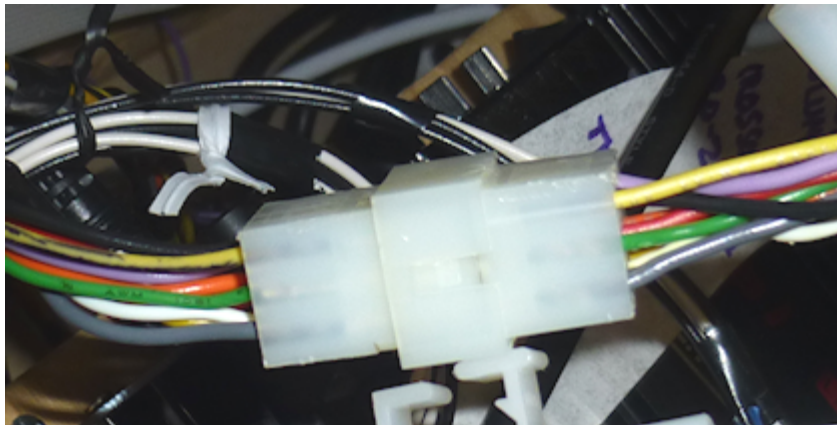
See Chapter 11, Power Switching for more on the front panel connector and how to set up PC-controlled power switching in your pin cab.

Molex wire-to-wire connectors

For any parts that you want to be able to easily remove from the cabinet, I always recommend using plug-in connectors. The pin headers described above are one type of plug-in connector, but those are mostly suitable for circuit boards. What about cases where you want to create a pluggable connection for something that's not a circuit board, like, say, your shaker motor?

For those connections, I like using a "wire-to-wire" connector. This is what they call any connector used to join two wire runs. A familiar household example is the connector at the end of an extension cord: it's there so that you can plug one wire (the power cord of an appliance) into another wire (the extension cord itself). Inside a pin cab, there are lots of situations where similar wire-to-wire plugs can be helpful.

There are lots of options for these. In my pin cab, I made lots of use of a couple of connector types made by Molex: their Standard .062" and Standard .093" systems. I was familiar with these because they're the connectors Williams used for similar purposes in their real machines for many years. The .062" and .093" systems are similar; the difference is the pin size, with the larger .093" pins having a higher power capacity. Both systems use plastic crimp-pin housings that come in a range of pin counts, with keyed connectors that only fit one way. This makes them pretty foolproof: the keying ensures that you can't plug a particular connector in the wrong way, and the different pin counts let you use a variety of different connectors in your cab to prevent mistaking one connector for another. You can't plug a 9-pin plug into a 12-pin socket.





Molex .062" 9-pin connector, plugged (top) and unplugged.

These systems are available as separate parts (male and female housings and crimp pins/sockets), but the easier way to buy them is in kits. You can find the kits on Amazon as well as Mouser. Here's a Mouser search that should work:

[www.mouser.com/Connectors/Pin-Socket-Connectors/_/N-ay0mm?
P=1z0yxz4](http://www.mouser.com/Connectors/Pin-Socket-Connectors/_/N-ay0mm?P=1z0yxz4)

(That's a search for Connectors > Pin & Socket connectors, product=Kits.)

Amazon searches for "Molex .062 kit" and "Molex .093 kit" should also turn up similar results.

These are crimp-pin systems, so see Chapter 82, Crimp Pins for instructions on assembling them.

Terminal blocks

An easy way to make point-to-point connections without soldering is with a screw-terminal block - a plastic block with a set of paired screw terminals, like the one pictured at right. Screw terminals are easy to connect and can be removed or changed at any time.



To make a connection between two wires with a screw terminal:

- Strip 1/4" or so of insulation from the end of each wire
- Pick a pair of connected screws on the terminal block
- Unscrew the screws a few turns (don't unscrew them completely)
- Wrap one of the wires around one of the screws, and tighten the screw to secure the wire
- Wrap the other wire around the other screw and tighten

Terminal blocks like this can be easily found on Amazon and eBay. I'd recommend getting the type that comes with plastic cover, which protects the terminals against accidental contact with other wires or metal parts.

Screw terminals are good for wiring that's more or less permanently installed in the cabinet. I wouldn't use screw terminals for any wiring that attaches to parts that you want to be able to remove frequently, because it's a bit of a pain to disconnect and reconnect the wires. It's better to use pluggable connectors, such as one the Molex wire-to-wire connectors mentioned above, for anything that you might have to remove more than rarely. For example, any parts that you have to remove to access other parts should be easily pluggable.

81. 0.1" Pin Headers

A very common type of connector used on printed circuit boards is the "pin header", so named because it has a series of metal pins that fit into sockets on a mating cable connector. The Pinscape boards use this type of header for most of their external connections.



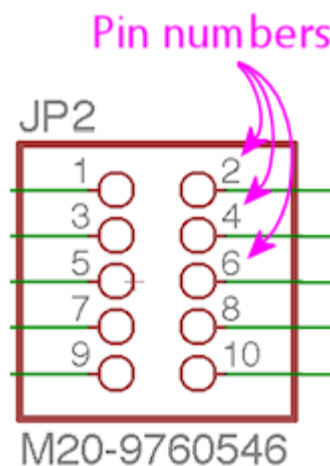
Pin headers in this style come in different sizes, but the most common (and the one used on the Pinscape boards) size is 0.1" pin pitch. This means that the pins are spaced at intervals of 1/10 of an inch. (To be more precise, that's the distance between the center points of adjacent pins, when viewing the header from directly above.) You'll sometimes see this expressed as 2.54mm pitch, which is of course the same as 0.1".

0.1" headers come in single and double rows. They're listed in a parts list according to the number of rows and columns: 2x09 for two rows of nine pins, for example.

Pin numbering

There's a standard convention for numbering the pins on the headers. If you have to look at the schematics, this will help you relate the wiring on the schematics to the physical pins on the board.

On the schematics, the pin numbers are indicated by little numbers printed next to the pins on the connector symbols:



On the Pinscape boards, the pin numbers aren't printed as such. Instead, there's a little white arrow pointing to pin #1:





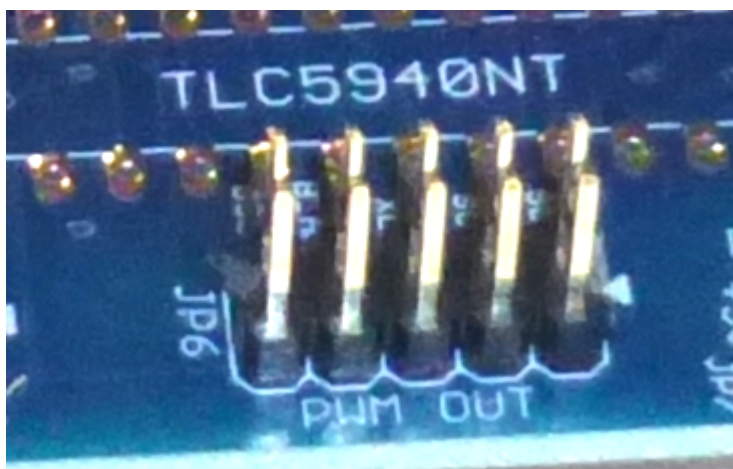
Once you've identified pin #1, you can determine the rest of the pin numbers just by counting them off starting at pin #1. The numbering pattern is always the same:

- For a single-row pin header, the pins are simply numbered sequentially across the row
- For a double-row header, the pins are numbered in "column" order:



Installing in a circuit board

The pictures above show how these look before you install them on the board. The black part is a plastic base that holds the pins at the proper spacing. You'll notice that the metal pins stick out of both sides of the base. The **long** part at the top is the connector, where you attach the mating cable socket. The **short** part at the bottom goes through holes on the board and gets soldered into place.



To install in a circuit board:

- If using a breakaway strip, use needle-nose pliers to break the strip to the required number of pins
- Insert the short end of the pins through the holes in the top of the board
- The plastic base should sit flat against the board as pictured above
- Hold the assembly in place and flip the board over

- Solder the pins to the pads on the bottom of the board

Some people ask if it's okay to just solder a couple of the pins, since that should be enough to hold the whole assembly in place mechanically. The answer is no: you have to solder **all** of the pins. The solder isn't only there to fasten the pin headers mechanically - it also makes the electrical connection between the pin and the solder pad. Each pin needs its own separate electrical connection to its pad, so every pin has to be properly soldered.

What to buy

The common 0.1" pin headers are interchangeable with each other and with 0.1" connector plugs. Any of the following options should work for the Pinscape board and KL25Z headers.

Option 1: Individual parts. The obvious thing to do is buy each size you need individually. If the parts list says you need a 2x8 for one header and a 2x10 for another, you go to Mouser and order a 2x8 and a 2x10. Our parts lists give you an exact Mouser part number for each size individually, so you can just order the items on the parts lists if you want to go this route.

Option 2: Breakaway strips. Alternatively, you can buy breakaway pin strips, which are extra-long versions that you can break up into sections to get the various sizes you need. These have from 20 to 50 pins in each row, and they're designed to be broken into smaller pieces at any pin. The plastic bases are notched so that you can easily split them up by hand or with pliers. This is the "buy in bulk" approach, so it's usually a little cheaper than buying the pre-cut sizes individually. It also simplifies things a little by cutting down on the number of different parts you have to order and keep track of.

Product options:

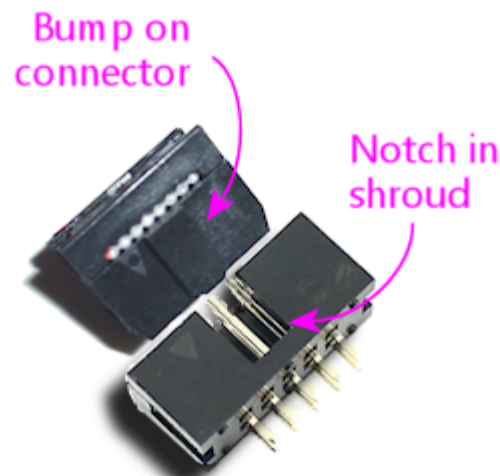
- Pololu has inexpensive unbranded single-row and double-row strips. These are under Electronics > Connectors > 0.1" Male Headers.
- TE Connectivity (at Mouser):
 - 5-826629-0 (1x50)
 - 5-826925-0 (2x50)
- Molex (at Mouser):
 - 90120-0160 (1x40)
 - 70280-0448 (2x50)

Shrouded pin headers



The plain pin strips pictured above are simple to set up, but they have the drawback that there's nothing to prevent you from accidentally plugging in the mating cable backwards. An alternative is to use "shrouded" headers. These have the same arrangement of metal pins as in the plain headers, and attach to a printed circuit board the same way, but they add a plastic box - the shroud - surrounding the pins.

The shroud offers some mechanical protection for the pins when the mating cable is unplugged, but its big benefit that it has a notch on one side (see the illustration at right) that accepts a corresponding "key" on a mating connector. The key is a little bump that fits into the notch; it makes it impossible to insert the cable backwards because the key won't fit in the other side where there's no notch.



Most IDC connectors (for Chapter 83, ribbon cables) have these keying bumps. Crimp housings sometimes have them and sometimes don't. Crimp housings that don't have the keying bumps will fit perfectly well into keyed shrouds, so there's no compatibility problem, but they'll fit backwards as well as forwards since they don't have the bumps to prevent backwards insertion. So you can use them, but you don't get the orientation sensing of the keyed kind.

The pins in a 0.1" shrouded header are identical to the pins in a 0.1" plain header, so you can often substitute a shrouded header anywhere a plain header is called for, or vice versa. However, the shroud takes up extra space, so there's not always room for it in a tightly packed board. Before substituting a shrouded header for a plain pin strip, make sure there's not another component (or another header) within the shroud's footprint on the top of the board.

The Pinscape expansion boards are designed to leave enough space around most of the data connectors to allow shrouded headers to be used.

Shrouded headers **won't** fit on the standalone KL25Z's pin pads. They're spaced too tightly.

What to buy

Molex 70246 series:

- 70246-0801 (2x04)
- 70246-1001 (2x05)
- 70246-1201 (2x06)
- 70246-1401 (2x07)
- 70246-1601 (2x08)
- 70246-2001 (2x10)
- 70246-2401 (2x12)

Matching connectors

A 0.1" pin header obviously only forms half of a connection. We now need something to plug into this header. There are two main types of matching connectors available:

- 0.1" crimp pin housings. A housing is the shell of a plug, which you have to build out with crimp pins. The idea is that you crimp a special little metal pin socket to a piece of hookup wire, and then insert the socket into the housing. You repeat for each pin position. When you plug the populated housing into the pin header, each pin socket mates with one pin on the header, forming a connection to the attached hookup wire.

Crimp pin housings are best when you specifically want to wire each pin separately, with a separate piece of hookup wire. This is appropriate when the far ends of the individual connections are scattered. For example, you'll want to connect the pins in the button input header on the Pinscape boards to the various buttons scattered around your cabinet, so a crimp pin housing is a good choice for that connector.

See below for more details on crimp pin housings.

- Ribbon cables. A ribbon cable is a flat cable with several conductors side-by-side. These are mated with IDC plugs, which is a connector that attaches to a ribbon cable via some little teeth that puncture the insulation and grab the wires. It's easy to assemble custom ribbon cables with these connectors, because you just line up the connector on the cable and squeeze the teeth into place.

Ribbon cables are ideal when you want to connect two like pin headers on different boards. You don't have to mess around with a bunch of individual hookup wires, and the connectors are quick and fairly easy to install because they install for all of the conductors at once. This is the right choice for the Pinscape board-to-board data connectors, such as the PWM data connection between the main board and the power board.

See Chapter 83, Ribbon Cables for details on buying and building these cables.

0.1" (2.54mm) crimp pin housings

This is one option for connecting to the common 0.1" pin headers detailed above. It's ideal when you specifically want to connect individual hookup wires to the pins, rather than using a single combined cable like a ribbon cable. For example, this is perfect for connecting buttons to the Pinscape expansion board button input header, since it lets you run an individual hookup wire to each button switch. That's good for buttons because they're scattered around the cabinet; a combined cable like a ribbon cable wouldn't be convenient for that since you'd have to split it up into individual wires to route it.

A "housing" is the *shell* of a plug. It's a plastic shell that holds the metal prongs that make up the plug, but it doesn't actually come with the plugs pre-installed. It's up to you to install the prongs, which are more technically called crimp pins.

This all takes some specially designed parts and tools. The crimp pins are specially designed to fit the housing, and they're specially designed to attach to hookup wire by crimping. They have little flaps that fold around the wire and grasp it to form a connection. You have to use a special tool (called a "crimp tool", naturally) that folds the flaps into just the right shape.

What to buy

Each manufacturer designs its crimp pin housings to work with its own crimp pins. You shouldn't try to mix and match housings and crimp pins; if you buy a Molex housing, you must use the matching Molex crimp pins.

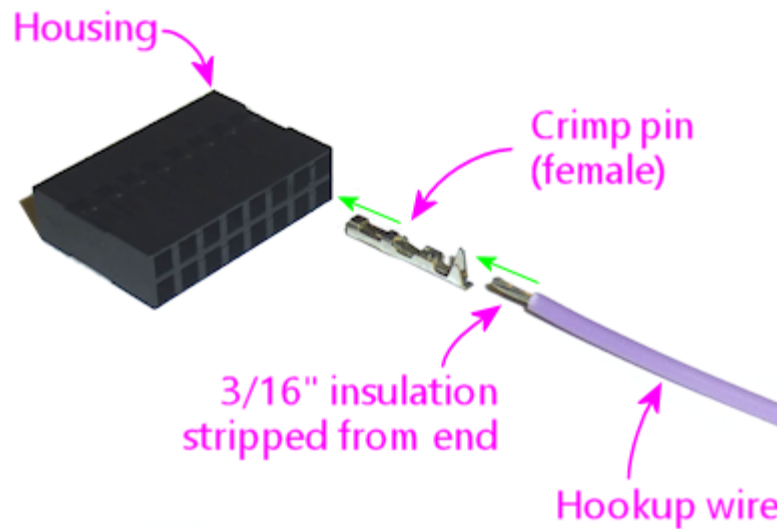
- Pololu has inexpensive crimp pin housings in sizes from 1x1 to 2x20 (although not in all possible sizes in between).
 - The housings are under Electronics > Connectors > Crimp Connector Housings
 - The matching crimp pins are under Electronics > Connectors > Female crimp pins for 0.1" housings (they're also linked from the pages for the individual housings)
- Molex makes housings from 1x2 to 2x32 under their CGRIDIII series:
 - Single-row housings: 90156-01XX (search for 90156-01 on Mouser for a list)
 - Double-row housings: 90142-00XX (search for 90142-00 on Mouser for a list)
 - Female crimp pins: 90119-0109 (for 22-24 AWG wire), 90119-0120 (for 26-48 AWG wire)
- Harwin makes housings from 1x2 to 2x12 under their M20 series:
 - Single row housings: M20-106XX00 (XX = number of pins, 02 to 12)
 - Double-row housings: M20-107XX00 (XX = pins per row, 02 to 12)
 - Polarizing key: M20-003
 - Female crimp pins: M20-116004X, M20-118004X (X = 2 gold/tin plated, 6 tin plated)

How to assemble

For instructions on assembling these connectors, see Chapter 82, Crimp Pins.

82. Crimp Pins

A crimp pin housing is a connector that you build yourself, by crimping hookup wires to individual metal pins, and then inserting the pins into the housing. The housing is essentially a plastic shell, with sockets that the pins lock into.



What makes crimp pin housings useful is that you get to attach your own wiring to the connector, with an individual hookup wire for each pin. That makes them ideal for connections where the individual wires won't all go to the same place at the other end. Some examples:

- The button input header on the Pinscape expansion boards. The header is set up for a single connector with 26 pins. Each pin connects to a wire that connects to an individual button, so these wires will all go off to different parts of the cabinet.
- The feedback device outputs on the Pinscape boards. Like the buttons, these headers provide a group of pins that plug into a single connector, and each pin on the connector will connect to a separate device.

Crimp pin housings are more work to assemble than ribbon cables with IDC plugs because of the pin-by-pin construction. That makes ribbon cables better for connections where all of the wires go off to the same place on the other end, such as a board-to-board data connection. Ribbon cables lose their convenience advantage when the wires all go to different destinations, because a ribbon cable is inherently constructed as a single unit with all of the conductors bundled together.

Tools

The key to easy assembly for these connectors is a **crimp tool**. This is a sort of hyper-specialized pliers that's purpose-built to perform the crimping step, where you attach the wire to the pin. If you have the right crimp tool, the crimping step is quick and easy.

This is the tool I use:

Pololu crimping tool, 16-28 AWG, item #1928

They also have a similar but slightly cheaper option, item #1929, which is basically the same tool with a more limited size range of 20-28 AWG. Either one of these will work with the parts needed for the Pinscape boards, but I'd spend the extra \$5 on

the wider-range model. That gives you the flexibility to use some of the larger Molex crimp connector systems elsewhere in the cabinet.

Assembly

We'll illustrate the assembly process using the Pololu 0.1" connectors. The other types of crimp connectors all work very much the same way, so you should be able to follow these steps for any of the other pin sizes and types.

This procedure is specifically for the Pololu crimper. The overall procedure should be the same for other crimper types, but I'm not sure they all use the same orientation for how you insert the pins. Check your crimper's instructions to see if you have to reverse anything in our illustrations.

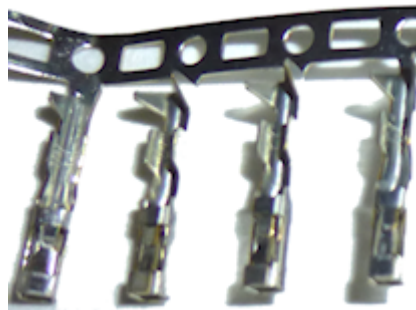
Step 1: Prepare the wire. Cut a piece of hookup wire to the desired length. Strip about 3/16" (5mm) of insulation from one end.



Make sure that the strands are nice and straight, and all tightly bundled. You can give them a little twist with your fingers if necessary to tighten the bundle. What you don't want is to have any stray strands sticking out sideways.

Be sure you're using the right wire gauge for the pin. Each type of pin has a compatible range of wire gauges, listed in its data sheet. The 0.1" type listed in the Pinscape parts list are good for wires in the 30-22 AWG range.

Step 2: Prepare the pin. Take the pin you want to crimp out of its packaging. The pins might come attached to a metal strip:



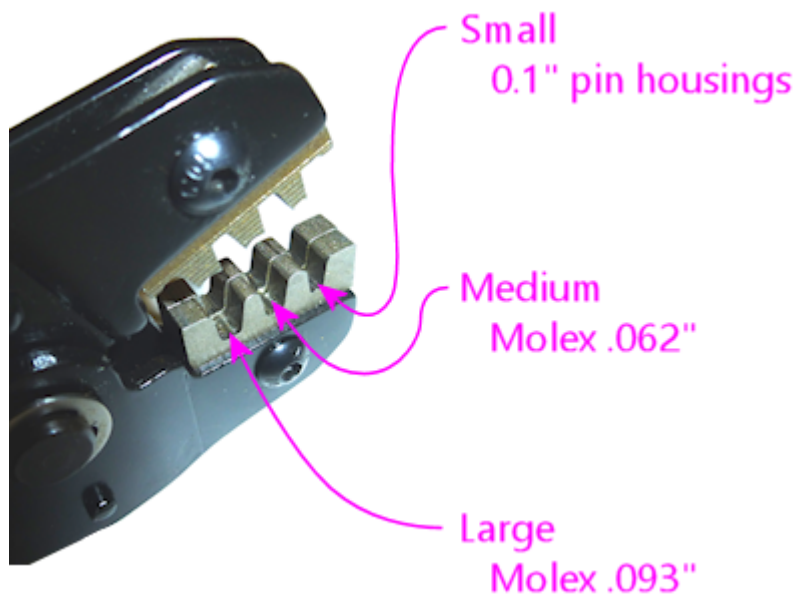
The pins are only attached to the strip by thin metal filaments that are meant to break away easily, so just bend it back and forth a few times to release it by way of metal fatigue.

Step 3: Prepare the crimp tool. If you're using the Pololu tool, make sure its jaws are full open. If they aren't, squeeze the handles all the way together and release. That should release the ratchet and let them spring open. If they lock shut, squeeze the handles even more tightly and release.

Orient the tool as shown below.



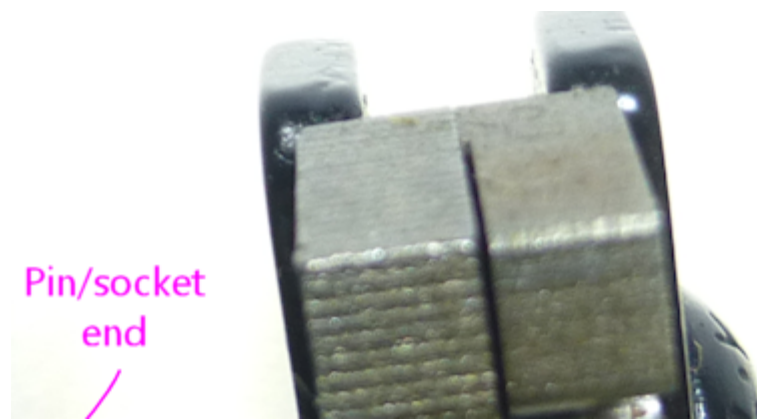
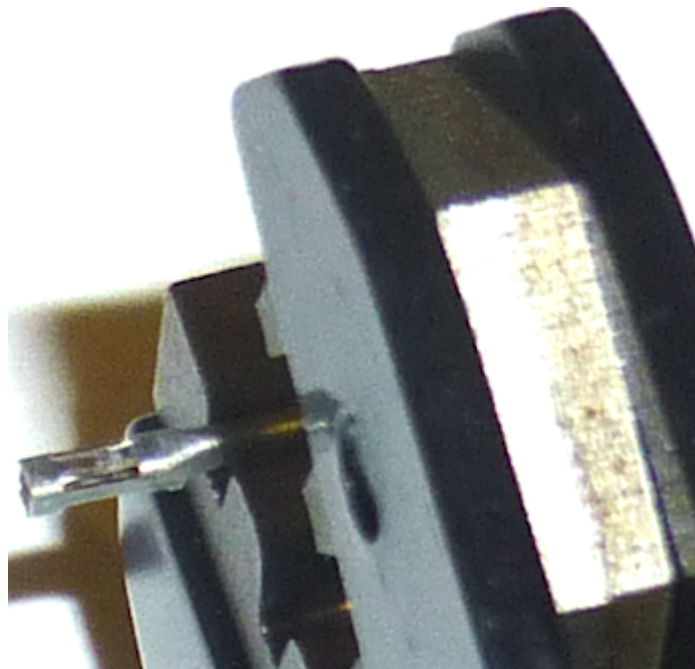
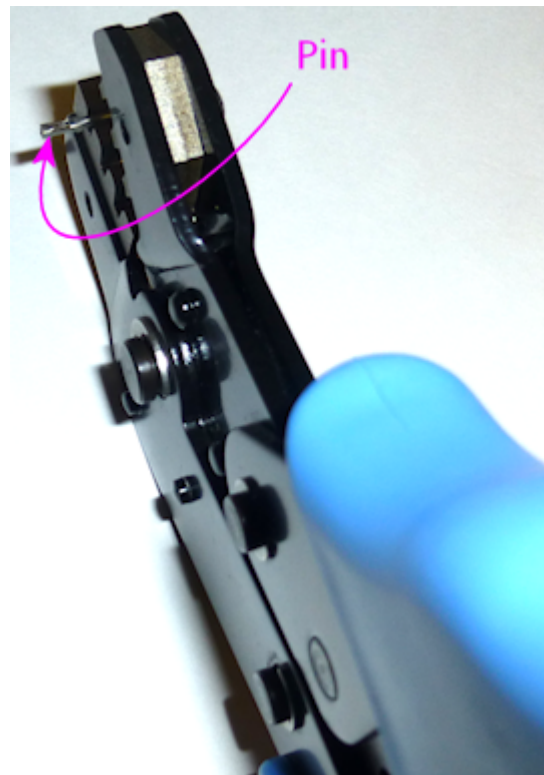
Step 4: Choose the slot. Choose the slot to use based on the size of the pin you're using.

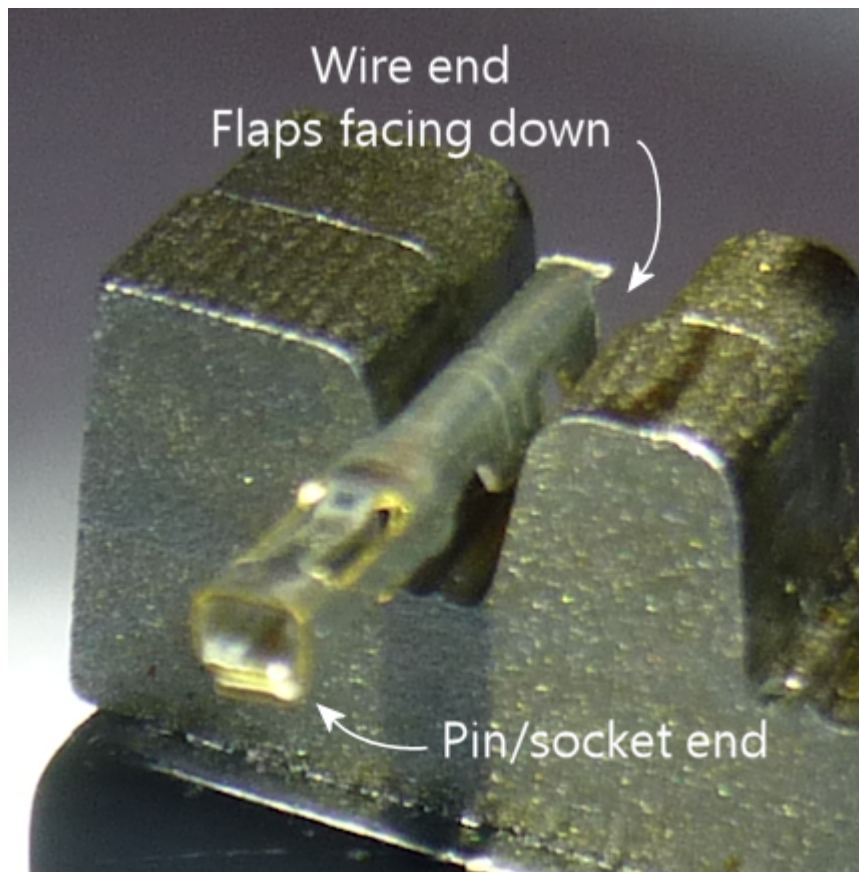
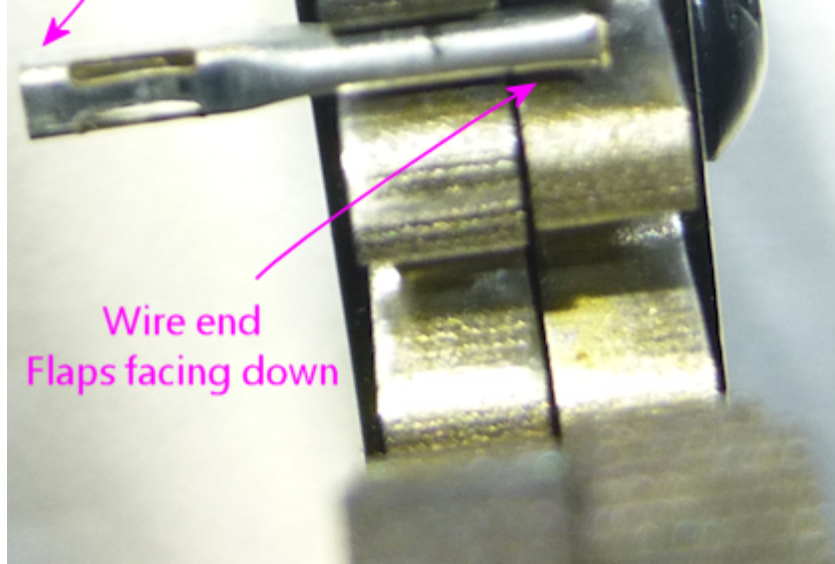


(The cheaper Pololu 2-slot tool has the same "small" and "medium" slots as the 3-slot tool.)

If you're not sure, try the slots from small to large. The pin should fit snugly when placed as in the next step.

Step 5: Insert the pin. Insert the pin on the **left** side of the slot as illustrated below. Be sure to use the correct slot for the pin size as shown above. Position the pin with the open side of the flaps pointing down.

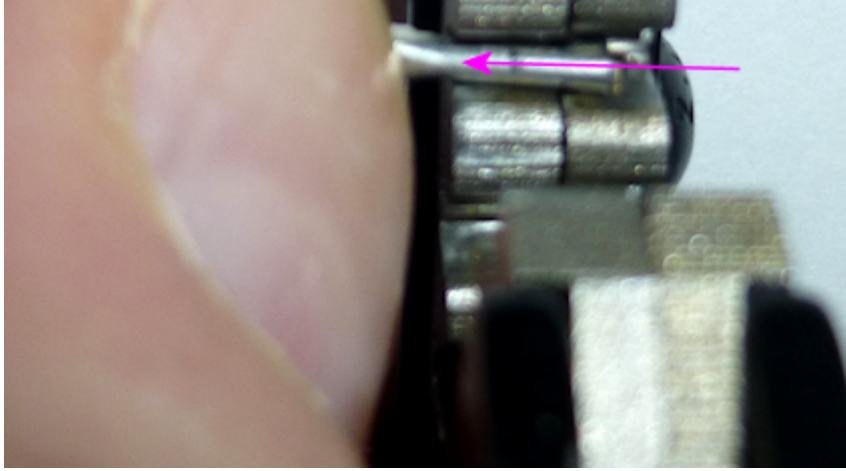




Be sure that **only** the part with the folding flaps is inside the teeth. The rest should be hanging out the left side.

Tip: you should be able to find the magic spot by feel. Hold the pin in the slot starting at the right side of the slot, and pull it left while keeping downward pressure on it. You should feel a little resistance when it reaches the right spot - this is where the larger outer flaps on the pin come up against the slightly narrower part of the teeth at the halfway point in the slot.



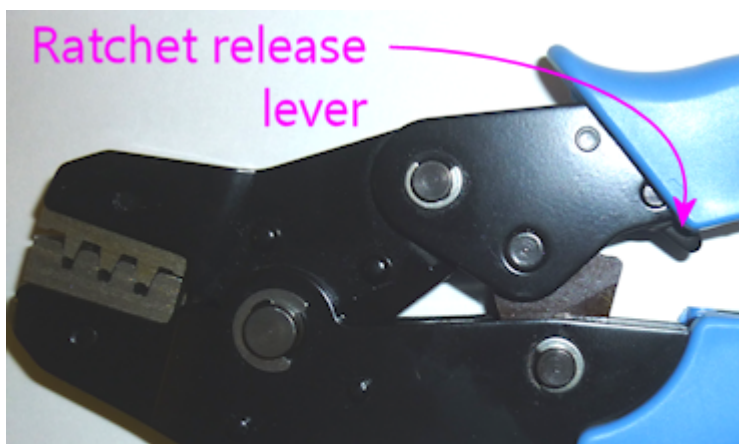


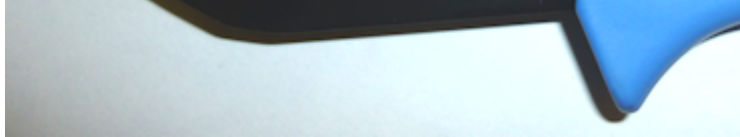
Step 6: Lock the pin. Lock the pin into place - but **don't** bend anything yet - by **gently** squeezing the handles until the teeth are closed *just far enough* to grasp the pin.



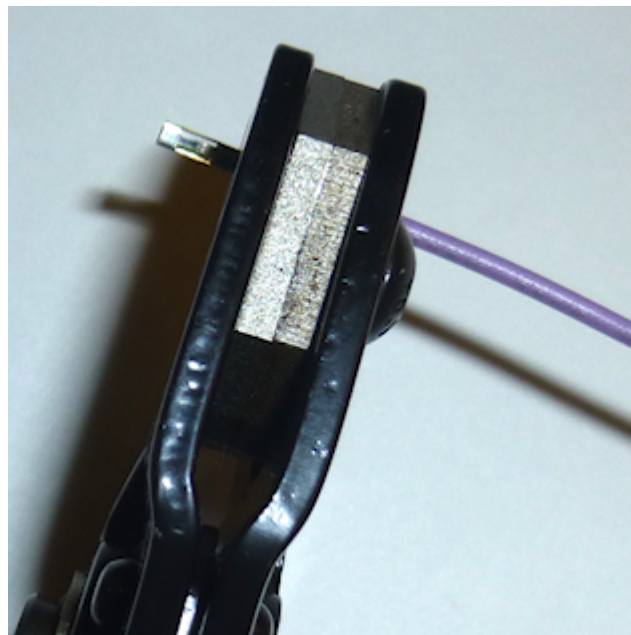
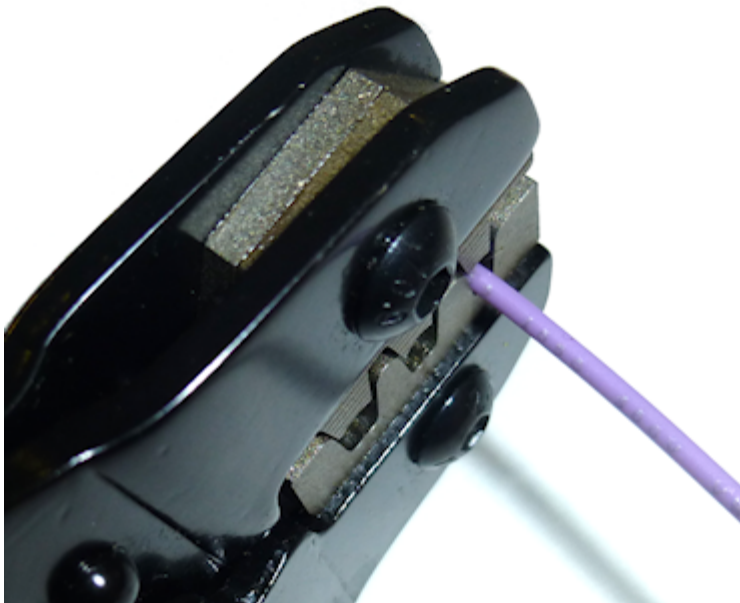
Remember, nothing gets bent at this point. The teeth should just be tight enough to hold the pin so that it can't move, but not any tighter.

Tip: if you got something wrong and you need to start over, you can release the ratchet early. **Very** gently apply just a little pressure to the handles, then press the little lever illustrated below to release the ratchet lock.

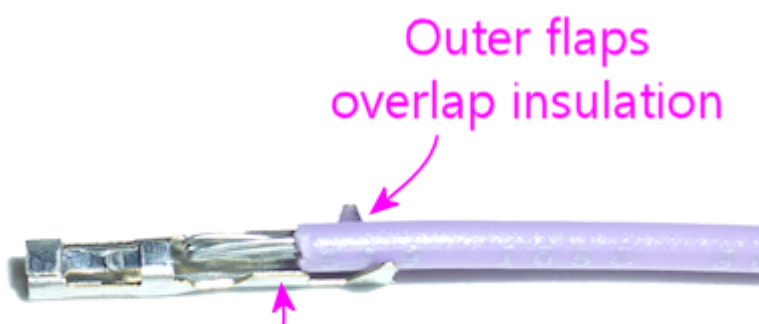




Step 7: Insert the wire. Insert the stripped end of the wire into the pin from the right side as illustrated below.



It's hard to see what you're doing with the pin held in the jaws, but here's how the wire is supposed to be positioned:

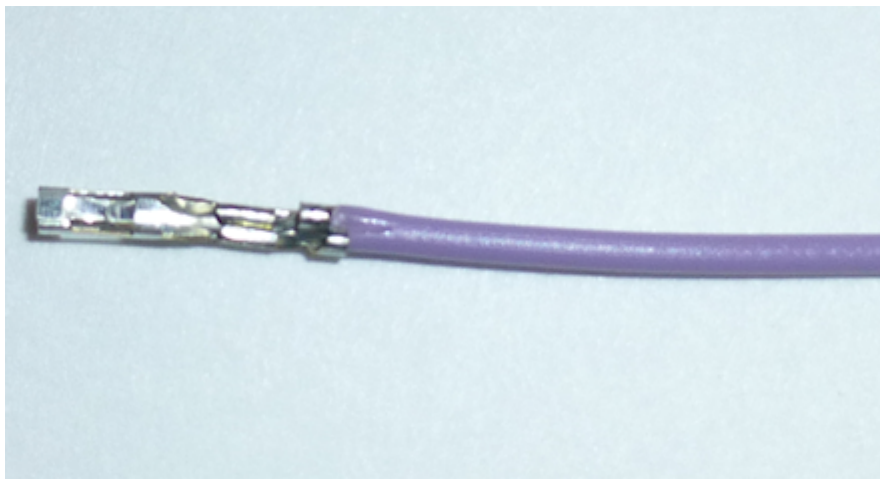


Inner flaps bare wire

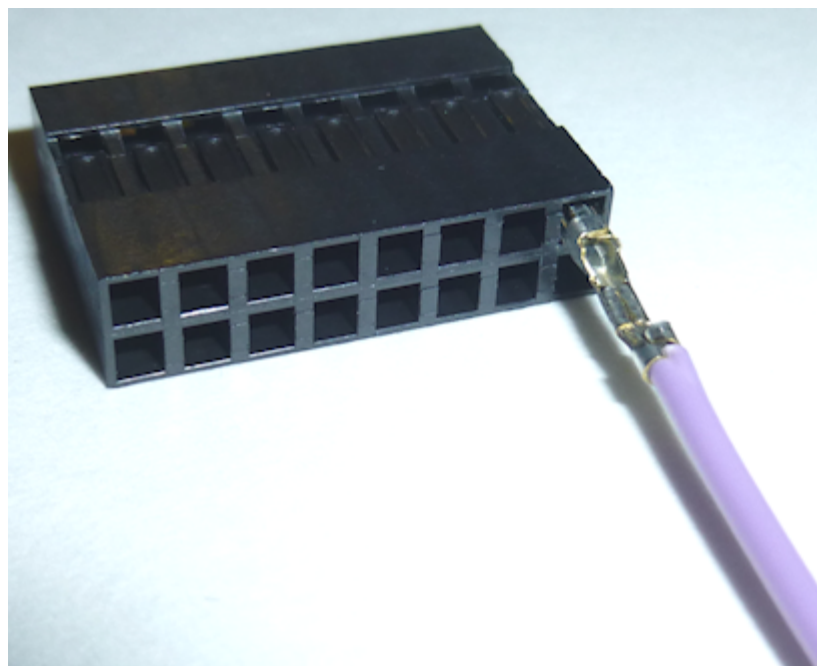
The idea is that the "inner" flaps grasp the bare part of the wire (for the electrical connection), and the "outer" flaps grasp the insulation (for a strong mechanical bond).

If you're using the right wire size for the pin, it should be easy to insert it to the right point by feel, because the insulation should easily fit through the outer flaps and then get blocked at the inner flaps. Make sure the wire strands make a nice straight bundle - any stray strands sticking out sideways can make it hard to find the right insertion point by feel.

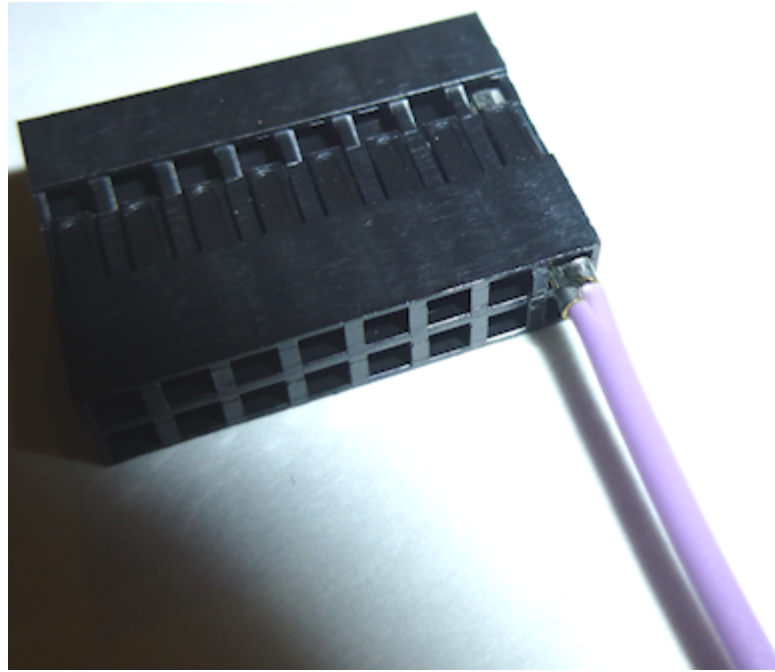
Step 8: Crimp! Making sure everything stays in place, squeeze the handles all the way together. That'll crimp the pin and release the ratchet. The finished wire-and-pin combo should pop out.



Step 9: Seat in housing. Find the slot where the pin goes in the housing. Insert it from the "back" side of the housing (the side the wires come out of). This is the side with the **larger** pin openings.



The pin should slide into the slot fairly easily. Push it all the way in until snaps into place.



If you can't get it to snap or lock into place, try taking the pin back out and rotating it 90°. The slot in the housing should have a little plastic latch that grabs onto a little tab sticking out from one side of the pin, so those have to align for the pin to lock. Each pin/housing system has its own peculiar mechanics for this (it's where the MEs at the connector companies earn their salaries, I suppose). Some of them work with the pin in any orientation, and some of them only work if the pin is lined up a certain way. It should be pretty obvious how the latch is supposed to work if you look closely at the pin and housing, but it's usually easy enough to do it by feel. I can usually get them on the second try if the first one didn't work.

83. Ribbon Cables

A ribbon cable is a flat, insulated cable with some number of conductors side by side, running parallel down the length of the cable. Ribbon cables are ideal when you want to connect two (or more) like connectors together. The flat geometry of the cable makes for good signal propagation in high-speed data connections, which is part of why these cables have long been used in the innards of PC wiring.

You can buy pre-assembled ribbon cables in certain lengths and widths (that is, the number of pins/conductors). However, it's hard to find them in anything other than a few common widths.

Fortunately, it's easy to build your own ribbon cables. You can get the bare cable in almost any length and width, and you can get the connectors in almost any pin count. Building your own cables is also cheaper than buying pre-assembled cables.

To build a ribbon cable, you need two types of parts:

- The ribbon cable itself
- IDC plugs, one for each end of the cable

"IDC" stands for Insulation-Displacement Connector. This is the type of connector normally used to build a ribbon cable. It attaches to the cable via little teeth that pierce the insulation and grab the wires, so all you have to install one of these connectors is position and squeeze it, which clamps it onto the cable.

Buying ribbon cable wire

The wire making up a ribbon cable is a generic part that you can buy anywhere. Mouser sells it by the foot, and you can also buy it on eBay, Amazon, and at Fry's Electronic.

Here's a Mouser search that will turn up some suitable options:

www.mouser.com/Wire-Cable/Cable-Assemblies/Ribbon-Cables-IDC-Cables/_/N-bkree?P=1y8vnl9Z1z0wxo9Z1yzi10l&Ns=Pricing%7c0

(That's a search for IDC ribbon cables, no connectors, 1.27mm pitch.)

Selecting wire: Ribbon cable is usually sold by unit length (foot or meter). Your vendor will probably have a giant spool and will cut the length you ask for. When figuring the required length, measure the distance between boards that the cable will have to cover, and add in some length for bends and to leave some slack. I usually double the apparent distance to make sure I have enough.

Apart from length, there are two main parameters when selecting cable: the number of conductors, and the wire "pitch" (the distance from one conductor to the next across the cable).

Pitch: For all of the expansion board connectors, you need **1.27mm** pitch wire.

In general, ribbon cables have **half** the pitch of the pin headers they connect to. The pin headers on the Pinscape expansion boards used with ribbon cable connectors are all the standard 0.1" pin pitch type, which equals 2.54mm. The matching ribbon cable pitch is half of 2.54mm, or 1.27mm.

Number of conductors: The number of conductors for a given connection has to match the number of pins on the header it connects to. For example, the plunger connector on the main expansion board (JP2) is an 8-pin (2x4) header, so it requires

an 8-conductor ribbon cable.

Splitting cables to make different widths: You can buy ribbon cables in a variety of widths (Mouser currently carries it in even numbers from 4 to 20 conductors), so you should be able to find it in each exact width you need.

It's not strictly necessary to buy the exact width, though, thanks to a little trick. You can easily tear a ribbon cable along the seam between any two wires to split it into two cables. For example, if you buy a 16-conductor cable, you can split it along the seam between the 8th and 9th wires to turn it into two 8-conductor cables. Or you can split a 24-wire cable in a 16-wire plus an 8-wire cable. The reason I mention this is that it's often cheaper to buy a single wider cable - a 20- or 24-conductor cable, perhaps - and split it up into the narrow widths you need.

Buying ribbon cable connectors

The specific connectors needed for the Pinscape boards are listed in the parts list.

0.1" ribbon cable connectors are standardized, so you can mix and match parts from different vendors for these. The Pinscape parts list uses parts from the series listed below (use the series name as a search key on Mouser to find the matching parts). These series include similar parts for different numbers of pins/conductors, in case you need to build cables for other pin header sizes.

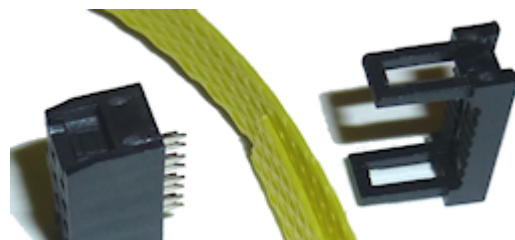
- Kobiconn 164-9000 series, available in various sizes from 10 to 50 pins
- Wurth Elektronik WR-BHD series, available in various sizes from 6 to 64 pins

Assembling a ribbon cable

Step 1: Prepare the connector. IDC plugs consist of two pieces that lock together around the cable, with the cable sandwiched between them. New connectors usually come with the two pieces loosely assembled but not locked, with a strip of cardboard or plastic to keep them separated until you're ready to assemble them.

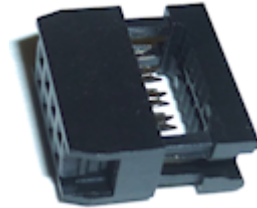


To prepare them for installation, you might be able to slip them off the end of the separator strip, or you might need to take the two pieces completely apart to get them free. To separate them, just pull them apart gently.

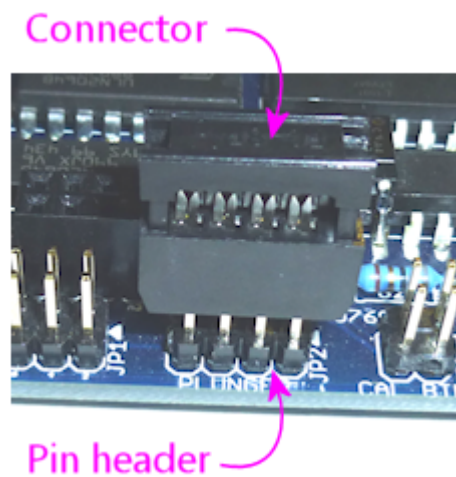


Be careful of the jagged metal teeth on the inside surface of the lower piece. They're razor-sharp by design, so that they can easily pierce the ribbon cable insulation when you install the connector.

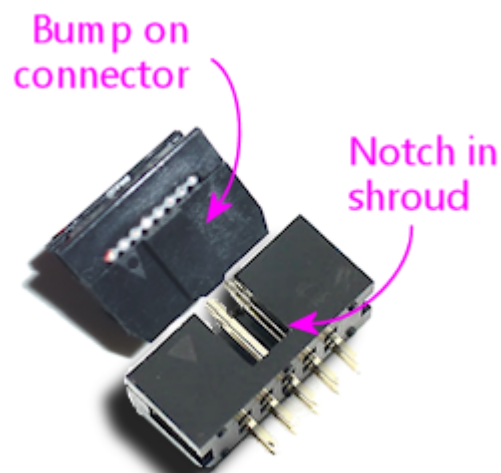
Once it's free, **loosely** reassemble the connector if you had to take it apart. **Don't** lock the two pieces together yet. Maintain as big a gap as possible between the two pieces. The cable has to feed into this gap.



Step 2: Position the connector. Fit the connector **loosely** onto the pin header where you'll ultimately plug this guy in.



If you're using a shrouded header on the board, make sure that the connector is oriented to match. Pay particular attention to the little bump on one side of the connector - it should match up with the slot on one side of the shrouded header. The bump and slot are the "key" that makes sure you can't plug the connector in backwards, so make sure they line up now.



Step 3: Cut the ribbon cable to the desired width and length. If you're using the "zip" method to split a wide ribbon cable into narrower cables, do so now to create a cable with the same number of conductors as there are pins on the connector you're attaching.

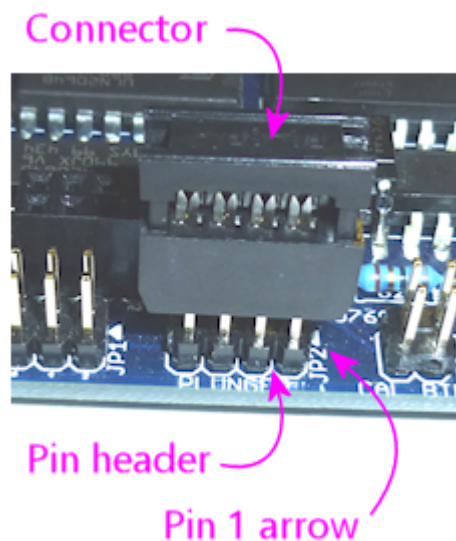
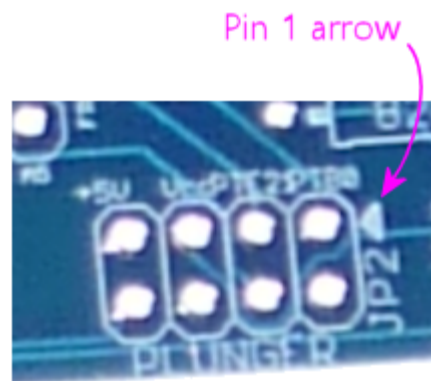
Cut the ribbon cable to the desired length. Most ribbon cables can be easily cut with scissors.

Step 4: Mark pin 1 on the cable. Use an **oil-based** red Sharpie to mark one edge of the cable as the "pin 1" side. You should use an oil-based marker because regular water-based inks won't stick to the plastic insulation.

The cable is symmetrical, so it doesn't matter which side you choose, but you should mark the cable down its whole length. This will help you orient the connectors properly by providing a visual marker to tell you where pin 1 is on the cable.



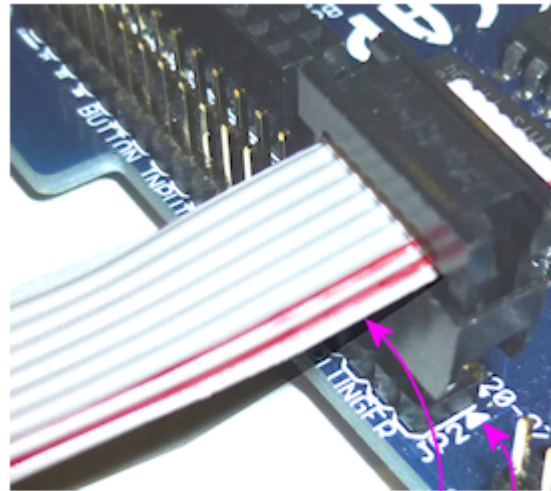
Step 5: Align the cable with the connector. Find the pin 1 mark on the circuit board next to the header where this cable plugs in. On the Pinscape boards, pin 1 is marked with a little triangular arrow at one corner:



Position the cable so that the red stripe you marked on the cable is on the same side as the pin 1 arrow.

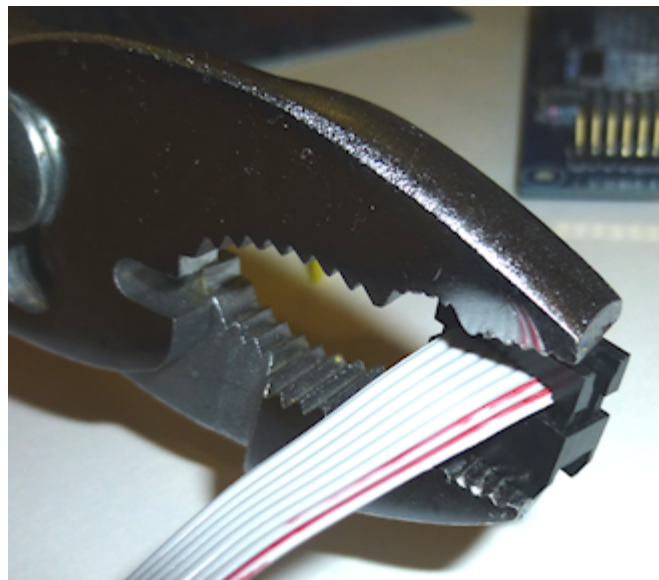
You'll usually want the connector to be attached near one end of the cable, so position the cable accordingly.

Step 6: Insert the cable. Keeping the connector and cable oriented as they are, insert the end of the cable into the gap in the connector. The cable should extend at least a little bit, perhaps an eighth of an inch, out of the other end of the connector. The cable should fit precisely across the width of the opening.

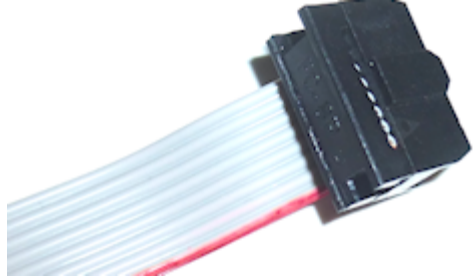


Red stripe
on same of connector
as pin 1 arrow

Step 7: Lock the connector. Take the connector off of the pin header, take care to keep the cable and connector pieces positioned as they were. Use an IDC crimp tool or a pair of pliers to squeeze the two halves of the connector together. Apply flat, even pressure across the whole width of the connector - it's important to press it straight down so that the teeth align properly with the individual conductors in the cable as they pierce the insulation.



Continue applying pressure until the two halves snap into place and lock together.



Step 8: Repeat for the other end. Repeat the whole procedure on the other end. Be sure to align the cable properly so that the side you marked with the red stripe for "pin 1" aligns with the circuit board marking for pin 1 on the other end of the cable.

Multiple connectors on one cable

One more thing to note about ribbon cables is that the IDC plugs aren't limited to the ends of the cable. They can also be attached at any desired intervals along the cable. This lets you create a single cable that connects three or more boards.

I use this feature on my own pin cab to connect two flasher panels with a single 16-pin ribbon cable:

- An IDC plug at one end of the cable plugs into the output controller
- A connector at the other end of the cable plugs into the flashers on top of the backbox
- A connector midway along the cable plugs into the flasher panel at the back of the main cabinet

84. Fuses

If you're including any feedback devices in your system, it's a good idea to include a fuse in each output circuit.

A fuse's job is to cut off power if too much current flows through a circuit. In your house wiring, the fuses and/or circuit breakers are there to prevent fires, by ensuring that your appliances don't try to force too much current through the wiring inside the walls. Wires that carry excessive current will overheat, and this obviously would be catastrophic for wiring inside walls.

Fuses help with fire protection in a pin cab context as well, but that's not usually our primary concern, because most everything in a pin cab draws its power from a DC power supply with its own built-in overload protection. The thing we're more concerned about in a pin cab is protecting our controller electronics against overload. Most controller boards have relatively low power-handling limits, typically well below the overload level of the power supplies, so it's good practice to use fuses to protect controller circuits from overload. This also has the side effect of giving us some additional protection against fire.

Are fuses required?

Fuses aren't absolutely required, but most pin cab builders consider them worthwhile. It comes down to a simple cost calculation: a 50-cent fuse might save a \$50 circuit board if an output overloads. I'd rather replace the 50-cent fuse than the \$50 circuit board.

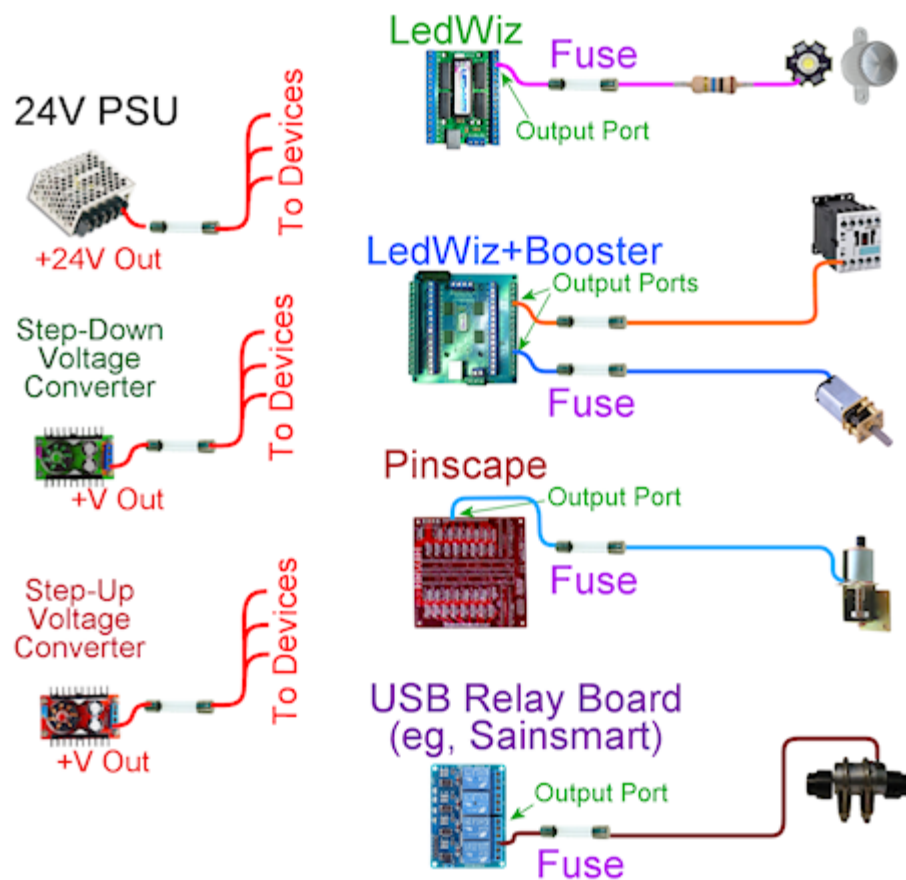
On the other hand, if you have a decked-out cab with twenty or thirty devices, the cost of parts for all of those fuses (taking into account the fuse holders and the additional wiring) might add up to more than the cost of the board you're protecting. So rather than fusing every output, you might prioritize the most "dangerous" outputs:

- Motors (including shakers, gear motors, and fans): very high priority. DC motors are particularly likely to cause random overloads because a motor's current draw increases with mechanical load. It's possible for a motor that's been working normally for a long time to suddenly draw a ton of current due to a mechanical issue, such as something jamming the motor.
- Solenoids: high priority. Solenoids can cause surges of current when first energized, and like motors, their current draw is somewhat unpredictable because it can depend on mechanical conditions.
- Contactors, relays: lower priority. These are basically solenoids at heart, but most have relatively low power needs and are self-contained enough that they're not likely to be affected by the mechanical environment.
- LEDs and incandescent lamps: low priority. These draw predictable current levels, and they're unlikely to misbehave at random. I'd skip fuses for these unless you're being very rigorous about fusing every last output.

Note that there's a secondary risk that applies to all of your output devices: wires that get crossed accidentally and create short-circuits. This can happen due to wiring errors, dropped screwdrivers, hex nuts that come loose and fall onto exposed contact points, etc. This isn't a purely hypothetical risk; these things actually do happen from time to time, judging by reports I've seen on the forums. This might be reason enough to use fuses for every feedback device.

Fuse wiring overview

We'll start with a diagram summarizing which devices need fuses, and where the fuses fit into the wiring. To make the diagram easier to read, we only show the wire segments where fuses are involved (so don't base your whole machine's wiring on this!).



Here are some key points to note:





- For feedback devices:
 - Each feedback device (contactor, flasher, motor, etc) should have its own fuse
 - Use fuses regardless of which output controller you're using
 - Wire the fuse between the output controller port and the device
 - Position the fuse close to the output controller port
- For **non-ATX** power supplies, and step-up/step-down voltage converters:
 - Use one fuse for the whole power supply or converter
 - Connect it to the **positive voltage output** (+V out)
 - Use the fuse as the common point of connection to all of the devices the PSU/converter powers
 - Place the fuse physically close to the PSU/converter
- For **ATX** power supplies (standard PC power supplies):
 - Fuses generally **aren't** needed for PC ATX-type power supplies, because these usually have built-in overload protection




- The built-in protection in an ATX supply is typically provided by a thermal "resettable fuse" that only cuts power temporarily, and automatically resets itself after the overheated parts cool off

Fuse selection quick reference

Here are some recommendations for fuses for the most common pin cab devices. For each device, we list the electrical specs you should look for, and provide an example of a fuse that meets the requirements. You don't have to use that exact fuse, just one that meets the specs listed. We'll explain what the specs mean and how we came up with them later in the chapter.

Fuses and ratings marked with asterisks (*) are examples only, because they're for component type that vary so much that no one fuse will be right for every example.

Device	Fuse Amps	Fuse Volts	Speed
 LedWiz output port (no boosters)	0.5A	$\geq 50\text{VDC}^{**}$	Fast
	Bel Fuse 2JQ 500-R		
 Zeb's LedWiz booster board output port	4A	$\geq 50\text{VDC}^{**}$	Fast
	Bel Fuse 2JQ 4-R		
 Pinscape power board output port	5A	$\geq 50\text{VDC}^{**}$	Normal
	Bel Fuse 2JQ 5-R		
 USB relay board (e.g., Sainsmart) output port	5A* (Use Max Amps for relay switch)	$\geq 50\text{VDC}^{**}$	Normal
	Bel Fuse 2JQ 5-R *		

 Replay knocker & other pinball coils	1.5A	$\geq 50\text{VDC}^{**}$	Slow- blow
	Bel Fuse 3JS 1.5-R TR		
 OEM/eBay/ generic power supply	10A* (Use PSU's Max Output Amps)	$\geq \text{PSU}$ output volts DC	Normal
	Littelfuse 0AGC010.V *		
 Step-up/Step- down DC-to- DC voltage converter	5A* (Use converter's Max Output Amps)	$\geq \text{Converter}$ output volts DC	Normal
	Bel Fuse 2JQ 5-R *		

* means that this is an example only, because this type of equipment varies. Check the actual Max Amps ratings on your equipment in these cases and substitute appropriate fuses if necessary.

** I used a blanket 50VDC recommendation for the sake of simplicity. This is high enough for anything commonly used in a pin cab, including knocker coils and other real pinball coils. It's actually difficult to find fuses rated for such high DC voltage, though; it's much easier to find 32VDC fuses, since that rating is used for virtually all automotive fuses. You can safely use 32VDC-rated fuses in any circuit where your actual power supply voltage is 32VDC or below.

How to choose a fuse

Fuses aren't one-size-fits-all. You have to choose fuses according to the electrical specs of the circuits they're protecting. Each circuit has its own requirements, so you might need different fuses for different circuits.

Current (Amps). This is the most important number when choosing a fuse. The whole purpose of a fuse is to limit the amount of current that's allowed to flow in a circuit, so choose a fuse for each circuit according to the maximum safe current for the components in the circuit.

A fuse's current rating tells you maximum the fuse will allow without blowing. For example, a fuse rated for 5A will allow up to 5A to flow, and should never blow as long as the current stays at or below 5A. If the current goes above the rated level, the fuse will blow - eventually. Not necessarily immediately. If the current is only a little above the limit, the fuse might not blow for minutes or even hours. The higher the current goes above the limit, the faster the fuse will

blow.

In a pin cab, most of our fuses are for protecting delicate electronics, like LedWiz outputs. These can be damaged very quickly by overloads, so we don't want the fuse to think about it for too long if the current exceeds the safe level for the device. A rule of thumb in these cases is to choose a fuse rated for about 75% of the maximum current for the device. For example, LedWiz outputs can be damaged above 500mA, so you might look for a fuse rated at around 375mA.

Voltage. The voltage rating on a fuse is a maximum. You don't have to match your circuit's voltage; you just have to make sure the fuse is rated for *at least* the circuit voltage. The voltage rating has nothing to do with the current limit, so it's fine to use a fuse with a higher voltage than your circuit uses. For example, a 125VDC fuse is fine to use in a 24VDC circuit.

You should select fuses with DC voltage ratings, since pin cab circuits are almost all DC. Many fuses are only rated for AC voltages. You might think this doesn't matter, but the fuse manufacturers warn that DC ratings are more stringent than AC, so a fuse that hasn't been rated for DC use might not properly stop current flow if used in a DC circuit. All of the fuses linked in this chapter are DC-rated.

Speed. Fuses come in two main speed classes: normal fuses that act quickly, and "slow-blow" fuses that act on a time delay. Slow-blow fuses are designed to withstand brief current overloads, for a period of several seconds to a couple of minutes, depending on how big an overload occurs. Normal fuses, in contrast, act quickly when the current goes over the limit.

Some manufacturers also make extra-fast fuses that operate even more quickly than the "normal" type, to protect especially sensitive circuits that can't tolerate even brief current surges.

The terminology for "fast" and "slow" can be confusing when shopping, because the terms aren't always used consistently. Some sellers use "fast" to refer to the extra-fast type, whereas others simply use "fast" for everything that's not "slow". When extra-fast fuses are in the mix, sellers will usually also offer "medium" or "normal" fuses, which refer to the regular fast-blow type, as opposed to extra-fast. If a site you're shopping at only offers "fast" and "slow" categories, you can probably assume that the "fast" ones are only the normally fast type.

Which speed class is best for a pin cab? It depends on the type of circuit you're protecting. For most circuits, normal fuses (not time-delayed and not extra-fast) will work well. You might consider extra-fast fuses in a few situations where small IC chips are part of the power circuit, specifically for LedWiz controllers, since the small driver chips on those devices can fail quickly when overloaded. Slow-blow fuses are useful for circuits driving pinball solenoids, such as replay knockers or chimes. Pinball coils are specifically designed to operate at intermittent "overload" levels, which is exactly what slow-blow fuses are good for, since they only act when an overload is sustained for an extended time period. We'll offer some advice on selecting slow-blow fuses for solenoid circuits in the section on coils below.

What if a fuse keeps blowing?

If a fuse in your cab blows repeatedly in normal use, one of two things could be going on. One possibility is that something's wrong with the circuit that's

causing an intermittent overload. The other is that you just need a fuse with a slightly higher limit.

To debug this kind of situation, I'd always start by assuming the worst - that there's a fault in the circuit that's causing a real overload situation. Carefully check the circuit for potential problems. If the fuse blows seemingly at random, check for the sorts of things that can cause intermittent problems, such as loose wires. If you can't find anything wrong, the next thing I'd do is double-check the current load of the circuit to make sure it's within the expected limits. For example, if this is a solenoid feedback device, use a multimeter to check the current drawn by the solenoid, and make sure that it's below the limit for the output controller port. If the device draws more current by design than the output controller allows, you'll have to either get a beefier output controller or substitute a smaller solenoid.

Assuming that you don't find anything wrong with the circuit, and that everything is within the expected limits, you might simply have to use a fuse with a slightly higher rating. For an inductive device like a motor or solenoid, if you're currently using a normal fuse, try switching to a slow-blow fuse with the same current rating. If you're already using a slow-blow fuse, try increasing the current rating on the fuse. Be conservative; raise it by maybe 25% of the original rating. Don't iterate this process indefinitely, though: it defeats the whole purpose of using a fuse if you have to use a fuse with such a high limit that something else in the circuit blows before the fuse does. If the problem doesn't clear up with a modest increase in the fuse rating, I'd go back and check again for faults, and failing that, I'd consider substituting a different device (a smaller solenoid, for example).

Form factor

Fuses come in several physical package types. For pin cabs, I recommend one of the types that plugs into a socket or holder, since this lets you replace a blown fuse by simply pulling the old one out of the socket and plugging in a new one. There are two main options for these: cartridge fuses and blade fuses.



I prefer cartridge fuses. They're the most widely used type in electronics in general, so they have the greatest range of options available.

Note that sizes for these fuses vary. There are about ten size classes each for the cartridge fuses and the blade fuses. The fuses linked in this chapter are all cartridge fuses in the 5x15mm (also known as 2AG) and 6.3x32mm (3AG) sizes. I apologize for not sticking to just one size: I was hoping to do that to keep things simpler, but unfortunately I wasn't able to find good matches for everything in one size, so I ended up with a mix.

Fuses holders

Cartridge fuses are designed to plug into sockets or holders, so you need the

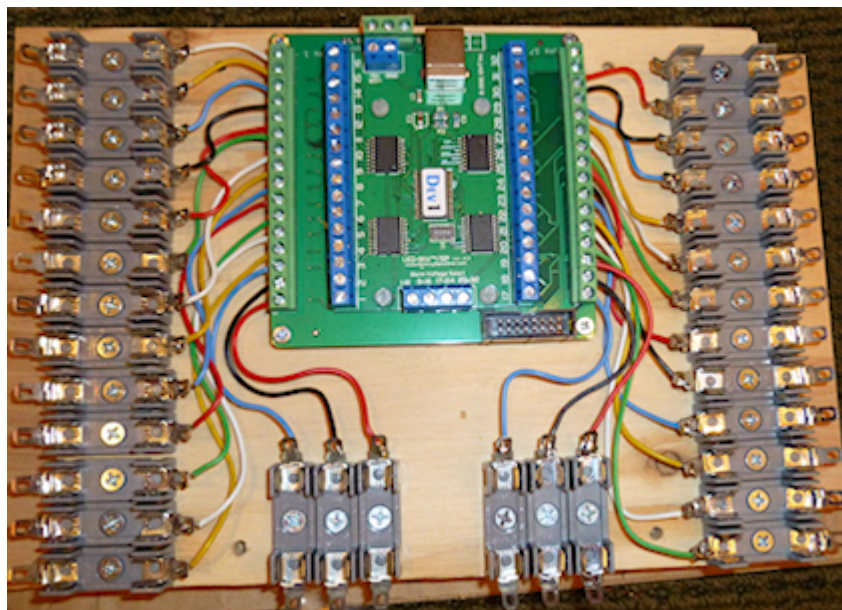
holders to complete your installation. There are several options available; search on Mouser or other electronics sites for "fuse holder". The type I like is a one-piece plastic holder, like these:



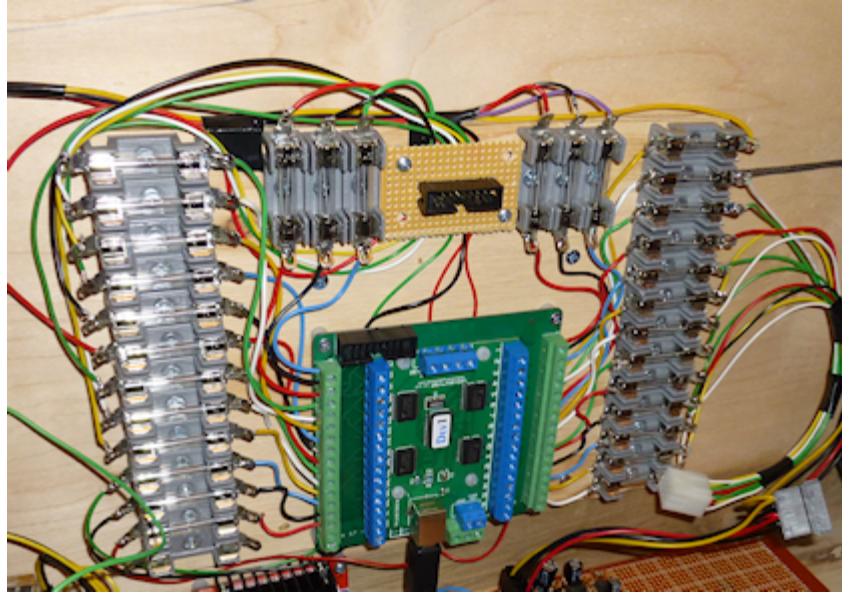
Note that there are several different sizes of cartridge fuses, so you'll need holders that match the size you're using. Here are example holders for the most common sizes (these cover all of the fuses linked in this chapter):

- 5x15mm (2AG)
- 6.3x32mm (3AG)

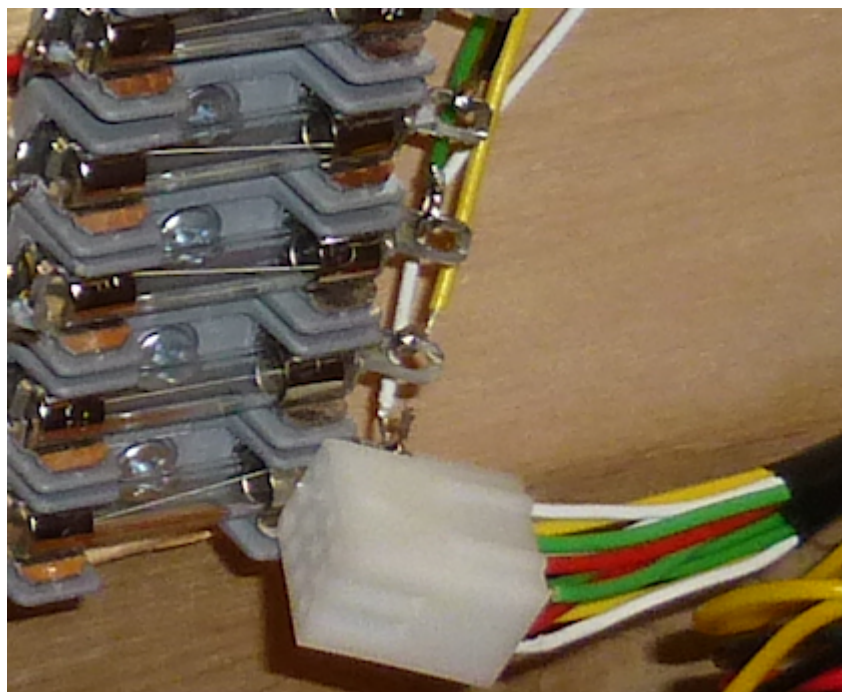
These have a screw hole for mounting to just about any surface, so you can mount them directly to your cabinet wall or floor or to a separate piece of plywood that you can later mount in the cabinet. If you're using an output controller, I recommend mounting the controller plus the fuses for all of its outputs on its own plywood carrier. That lets you do all of the initial wiring on your workbench rather than in the confined spaces inside your cab. If you're using an LedWiz or Pinscape board, you'll have a lot of fuses to install, so it makes the work a lot easier.



Once it's all set up, you can then mount the whole thing in your cabinet with a couple of screws. This also lets you remove the whole setup later if you ever need to do any repair or upgrade work.



Note that a key element of a modular setup like this is pluggable connectors. You'll want to use some kind of mating plug-and-socket connector to connect all of the wires coming out of the fuses to the devices inside the cabinet. You can see one of the connectors I use in the lower right of the picture above; here's a closeup.



That's a connector from the Molex .062" series, which I found useful for many of the interconnects in my cabinet. There's more information on these and other options in Chapter 80, Connectors. The thing I like about pluggable connectors like this is that they help avoid dumb mistakes. You only have to plan out and wire the connectors once, and from then on it's just a matter of plugging the mating connectors back together any time you have to do any work that involves removing something. You don't have to match up the individual wires again, since they're bundled into connectors that only fit one way.

What to protect

If you're not experienced with electronics (and even if you are), it can be tempting to add fuses anywhere and everywhere. But every fuse you add has costs: not just the monetary cost of the fuse, but the space it takes up, the time it takes to wire, and the additional point of failure. It's best to limit yourself to circuits that really benefit from fuse protection.

Let's look at the places in a typical pin cab where fuses are most useful.

Output controllers

If you're using any sort of output controller to attach feedback devices (such as solenoids, contactors, lights, motors, etc.), it's a good idea to place a fuse in each individual output circuit. This is probably the most important place to use fuses in the whole cab, because it's the place where things are most likely to go wrong.

In this case, the point is to protect the **controller**. We're protecting the controller from two things. First, from simple short circuits. Feedback devices are scattered around the cabinet, so they often have long wire runs, leaving lots of openings for accidental shorts. Second, we want to protect the controller from the attached device. Every output controller has limits on how big a load it can handle, so we want to make sure that the attached device doesn't draw too much power, either routinely or due to a malfunction. A fuse accomplishes that by shutting down the circuit if the power draw goes over the limit.

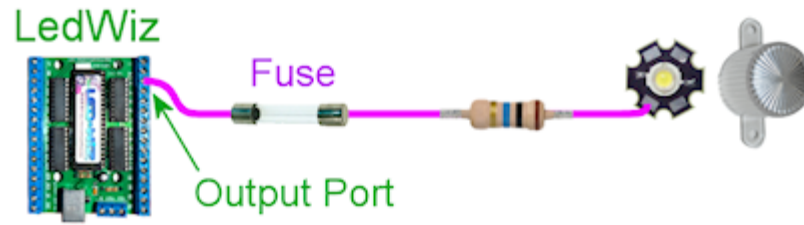
In most cases, we don't need to be concerned with protecting the feedback device itself (the light, motor, etc). That device is usually the threat, not the victim. If anything else in the circuit malfunctions, the worst that usually happens as far as the device is concerned is that it gets turned on at full power. But that's what it's designed for in the first place, so this usually isn't a threat. The exception to this rule is pinball coils, which we'll come to shortly.

Since we're protecting the output controller, we need to choose a fuse based on the current limit of the controller:

- LedWiz (with no booster board): 500mA (0.5A) per output.
- Pac-Drive (with no booster board): 500mA (0.5A) per output.
- Pinscape power board: 5A per output.
- Pinscape main board flasher & strobe outputs: 1.5A per output.
- Pinscape main board flipper button LED outputs: no fuses are necessary, because these outputs have built-in current limiters.
- Pinscape DIY MOSFET output circuit: the limit depends primarily on MOSFET you choose, but 5A is a safe (conservative) choice for all of the options we recommend in our circuit plans.
- Zeb's booster board for LedWiz: 5A per output. You **don't** need a separate fuse for the LedWiz in this case, because the booster board isolates the LedWiz.
- PacLed, i-Pac Ultimate I/O: these have built-in current limiters per output, so fuses aren't required.
- Zeb's booster board for PacLed: 5A per output. You **don't** need a separate fuse for the PacLed.
- USB relay boards (e.g., Sainsmart): Check the specs for your board for the DC current limit for the relay switch. It's usually about 10A.

The wiring for a fuse in an output circuit is the same for all of the controllers,

so we'll just show the LedWiz as an example. The basic plan is to interpose the fuse between the output controller port and the device.



First, connect a wire between the output port on the controller and one end of the fuse. Next, connect a wire from the other end of the fuse to the feedback device (the light, motor, etc). If the device cares about polarity, this is the **negative** or "ground" terminal on the device.

In the diagram above, we used an LED as the example output device, so there's actually an extra step involved, because LEDs usually require resistors (that's a whole separate subject, explained in Chapter 44, Feedback Devices Overview). In this case, the resistor goes between the fuse and the LED, so we connect the wire from the fuse to one end of the resistor, and connect a wire from the other end of the resistor to the output device. For anything but an LED, there's no resistor, so the wire from the fuse goes straight to the device.

Fuses don't care about polarity, so it doesn't matter which direction it goes. Resistors don't care either.

Pinball coils

Unlike most other feedback devices, real pinball coils can benefit from fuse protection. As we mentioned above, most other feedback devices don't need fuses for their own sake; the fuse is to protect the output controller, not the device itself. But pinball coils are different. They're designed to be activated only in split-second bursts. If you turn one on and leave it on, it'll quickly get so hot that its internal wiring melts, destroying it.

To protect against this danger, we can use a special type of fuse called a "slow-blow" fuse. "Slow-blow" means that the fuse takes longer to blow than a regular fuse does. A slow-blow fuse allows a surge of current to flow briefly, but if the current is sustained for too long, the fuse blows. This is exactly what we need to protect pinball coils, which are likewise designed for brief bursts of power, but can't withstand sustained use.

Why are we even worried about this? We only use these coils to simulate bumpers and other things that fire momentarily, so why would the software ever leave them on for long periods? Normally, it wouldn't. The danger is that the software doesn't always work perfectly. Sometimes it crashes, and if it crashes at the wrong moment, it can leave an output stuck on. This isn't just a theoretical possibility, either: this has actually happened to other cab builders. It might seem incredibly improbable that the software would crash at such a perfectly wrong moment, but it's actually an especially likely time to crash, because it takes special code to fire a coil in the first place. That code can contain an error that makes the program crash right after the coil turns on, so the code that was supposed to turn the coil back off never gets a chance to run, leaving the coil stuck on.

As an alternative to using fuses, or as a second layer of protection, you can use a special time limiter circuit. These circuits contain their own timers that turn the coil off after a couple of seconds, even if the software doesn't. See Chapter 54, Coil Timers for details on how to build one of these. If you're using a Pinscape main expansion board for your knocker, it has this type of timer built in to the knocker output. All of the outputs on the Pinscape chime board also have these timers. In my own cab, I use both the timer and the fuse for my knocker coil. I think of the timer as the primary protection, but I still like having a fuse as a last resort in case the timer fails.

The types of pinball coils that can benefit from fuses include:

- Replay knockers
- Chime units
- Bells
- Bumper assemblies
- Slingshots

Flippers are more complicated; more on them below.

How do you choose the right slow-blow fuse to protect a pinball coil? It takes a little research and some guesswork.

The first step is to figure the normal operating current for the coil.

You need two numbers to figure the current: the voltage you're going to use to operate the coil, and the coil's electrical resistance. The voltage is easy: it's the voltage of the power supply you're going to use with the coil. The resistance is something you can measure with a multimeter: set your meter to the "resistance" or "ohms" setting and measure across the coil's terminals. You should measure the coil while it's not connected to anything else, of course. Here are the specs for some commonly used knocker coils:

- AE-26-1200: 10.9 ohms
- AE-23-800: 4.2 ohms

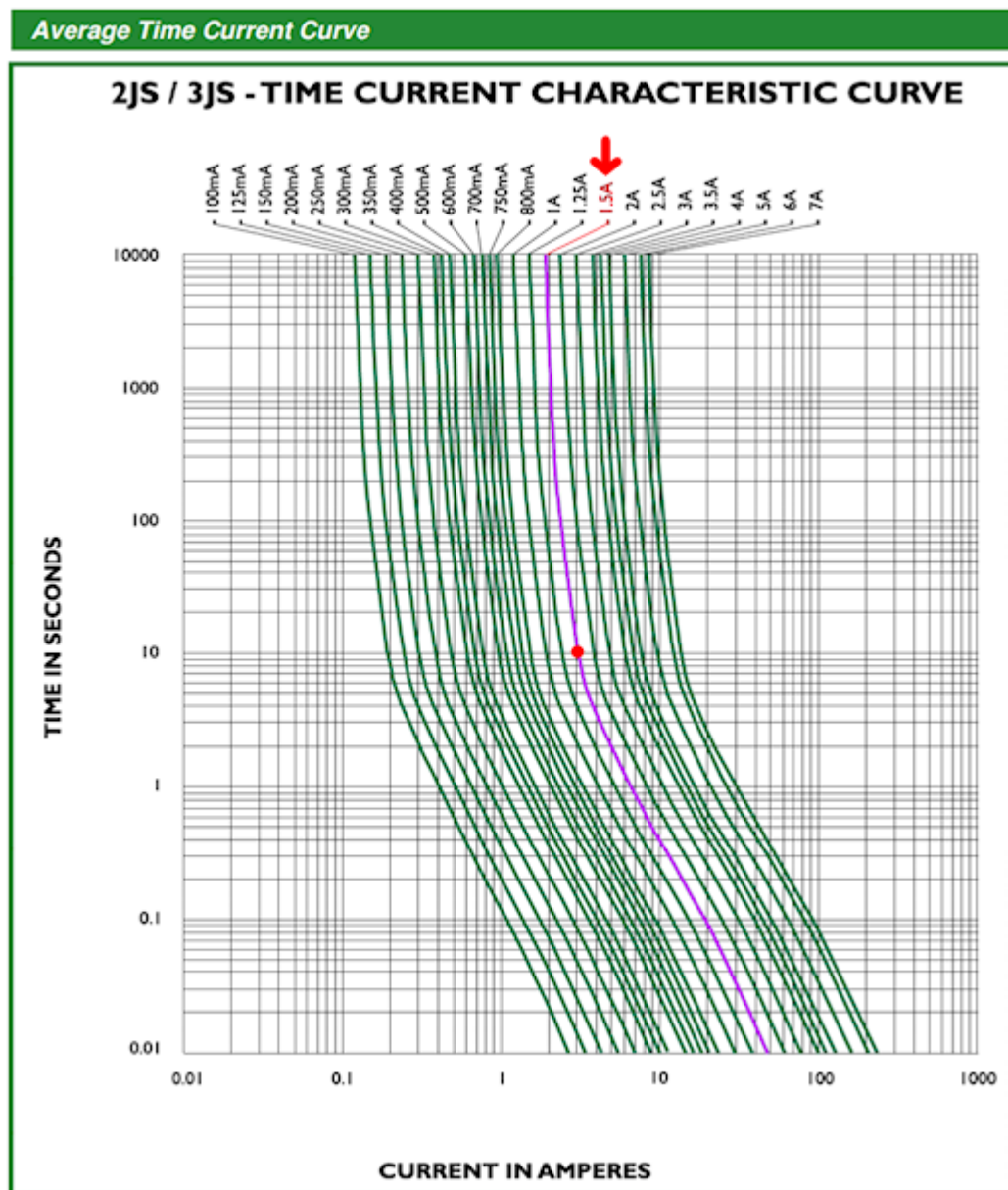
Once you have the voltage and resistance, determine the amperage as Volts/ Ohms. For example, if you have a replay knocker with an AE-26-1200 coil that you'll operate at 35V, the current is $35V/10.9\Omega = 3.2A$.

Step two is to decide on a time limit. This is balancing act. We want to pick a time limit that's short enough that the fuse will blow before the coil overheats, but long enough that the fuse won't blow during normal operation. The complication is that slow-blow fuses have inexact timing. They don't give you an exact delay time, just a range of times.

Pinball coils on real machines fire for a fraction of a second, anywhere from a few milliseconds to about half a second. We can consider times in this range to be safe for the coil, so we don't want our fuse to blow within the first half-second. But how long is too long? Unfortunately, I don't have any hard data on that. I haven't been willing to sacrifice a bunch of coils to methodically measure burn-out times experimentally, and as far as I know, neither has anyone else. So this is where a bit of guesswork comes in. Anecdotally, I've heard from a few people who've had their coils get stuck on and burn out. From those reports, it appears that coils will pretty reliable overheat after about a minute, maybe two. I've also heard one or two reports of failure after much shorter times, around 10 seconds. Based on these reports, it seems best

to choose a fuse that will blow after perhaps 10 to 20 seconds.

Step three is to choose a fuse that matches the combination of the current and time we came up with in steps one and two. This is another research step, because slow-blow fuses aren't sold with simple, fixed time limits. Instead, the time limit is a function of the current. The manufacturer presents this function with a chart in the data sheet, like this one, taken from the data sheet for the Bel Fuse 3JS data sheet:



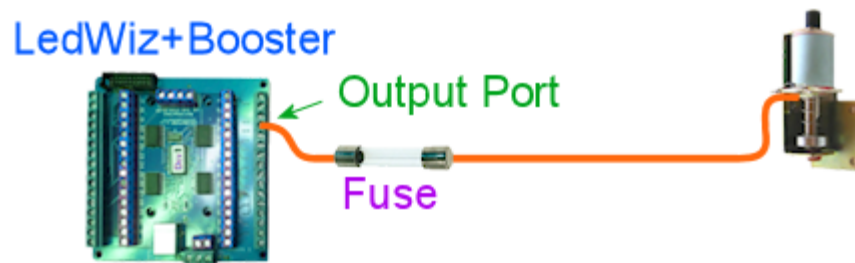
Each green curve represents the current/time relationship for one type of fuse, labeled at the top.

Here's the way I use this type of chart to pick a fuse. Let's say we want to pick a fuse for a replay knocker with an AE-26-1200 coil that we're running at 35V. As we calculated above, that gives us 3.2A as the normal operating current for the coil, and as we guesstimated above, we'd like a fuse that blows in perhaps 10-20 seconds. So let's go to the chart, and find the intersection of 10 seconds and 3.2A. I marked that spot with a red dot. That happens to fall right on one of the green lines - if it didn't, it would be between two lines, so we could pick the closest. I highlighted the line we're on to make it easier to follow. Now trace the line to the top of the chart to see which fuse it's for: it's the 1-1/2A (1.5A) fuse.

It seems a little strange that we're going to use a 1.5A fuse for a 3A coil, but the amp rating on the fuse is only nominal. The timing chart tells the full story, and in this case the timing chart says that the nominally 1.5A fuse will let our 3 Amps flow for up to about 10 seconds. In case you're still not convinced, I've been running with this fuse protecting my own AE-26-1200 replay coil, and it hasn't ever blown on a false alarm.

Since this is such a common coil in virtual pinball machines, I'll save you the trouble of finding the 1.5A version of this family. Here's the part number and a Mouser link: Bel Fuse 3JS 1.5-R TR.

Wiring the fuse for a knocker is just like wiring a fuse for any other output device. The fuse goes between the output port on the controller board and the knocker coil. Here's a diagram; we use an LedWiz with booster board as an example, but it's the same for any other type of output controller.



First, run a wire from the output port on the controller board to one end of the fuse. Next, run a wire from the other end of the fuse to the knocker. This connects to the **negative** side of the knocker coil, usually signified by a black wire. (The coil itself doesn't care about polarity, but it should have a diode attached, and the diode does care.) Fuses have no polarity, so the direction you connect the fuse doesn't matter.

Flipper coils

Flipper coils are more complicated than most other types of pinball coils, but in a way that actually simplifies things for our purposes here. In most cases, you won't need a fuse for a real flipper coil. The reason is that flipper coils for real machines are built to tolerate being activated for long periods. They have to be, because players routinely hold a flipper up to trap a ball. So unlike other pinball coils, these coils are designed to withstand continuous power, and thus don't need to be protected from getting stuck on.

How do they accomplish this, when other coils can't? Their trick is a clever two-coil design. Flipper coils are really two coils in one: two separate coils of wire wrapped around the same core. One coil is the high-power "lift" coil, which generates the strong initial force that lifts the flipper from the rest position and propels it (and the ball) through the flipper's arc. The other coil is the low-power "hold" coil, which only kicks in when the flipper reaches the top of its arc. The hold coil only has to exert enough force to hold the flipper in place; it doesn't have to propel the flipper or the ball. This allows the hold coil to operate at much lower power - low enough that it can be left on indefinitely without overheating. The flipper assembly toggles power between the two coils by way of an end-of-stroke switch, which the flipper trips mechanically when it reaches the top of its arc.

If you're planning to use a real flipper coil in your virtual cab, you should make sure that it's part of a full flipper assembly that has the end-of-stroke switch in

place and properly adjusted. The end-of-stroke switch is critical because it's what prevents the lift coil from getting stuck on. If the lift coil gets stuck on, it'll overheat like any other pinball coil.

The dual-coil design isn't universal. Newer Stern machines (2000s onward) dispense with this somewhat complex electro-mechanical design and use a somewhat complex software system instead. On these machines, the flipper coil is just an ordinary coil with a single winding. The flipper button is connected to the CPU, not directly to the flipper. When you press the button, the CPU turns the coil on at full strength. But this lasts only for a split second, long enough for the lift stroke. Once that initial time period has passed, the software reduces power to the coil using PWM, or pulse-width modulation. PWM is a method for controlling power by switching the voltage on and off rapidly (hundreds of times a second).

The single-coil Stern design isn't suitable for virtual cabs, because it requires the specialized software system to manage the power. None of the current software or hardware in the pin cab environment can do this. So if you want to use a real flipper assembly, you should use the traditional dual-coil type, not the newer Stern single-coil type.

Other solenoids

Solenoids might or might need the same protection as pinball coils (see above) against being left on for long periods.

To find out whether you need a fuse for your particular solenoid, start with the data sheet, if one is available. Look for information on maximum continuous "on" time.

If you can't find a data sheet, or it has nothing to say on the subject, you can do some testing of your own. Apply power to the solenoid for a couple of seconds, then cut power. Check to see if the solenoid feels hot. If not, try again, leaving it on a little longer. Repeat, extending the time on each trial, until the solenoid starts feeling hot to the touch. If you can leave it energized continuously for several minutes without it getting too hot, you probably don't have to worry about a special fuse for it. If it does start getting hot within a couple of minutes, though, you should add a slow-blow fuse using the same procedure explained above for pinball coils.

If you're using a solenoid to simulate any sort of rapid-fire device, like a bumper, slingshot, replay knocker, chime, bell, etc., the same rule of thumb for timing that we used for pinball coils should work well here. However, you might want to extend the maximum time a bit, like taking it up to 30 to 60 seconds, assuming your solenoid didn't overheat that quickly in the tests above. The reason is that bumpers and the like get a lot of use in some games - they fire briefly, but many times in a row. Many short bursts in a short time add up, since this is all about heat dissipation. So choosing a fuse with too short a time limit might result in the fuse blowing unnecessary during bursts of activity during normal play.

Wire the fuse for a solenoid the same way you would for a pinball coil.

PC/ATX power supplies

Good news! You probably **don't** need any fuses for PC ATX power supplies. These almost always have built-in thermal overload protection that temporarily shuts them down if they overheat. That's the same function a fuse performs, so there's no need for a separate fuse; the built-in protection is all you need.

The thermal protection circuit in an ATX power supply should automatically reset itself when the temperature returns to normal, so you won't have to open it up to replace a fuse, or even push a reset button, if you ever trigger an overload. Just unplug everything for fifteen minutes or so to let the power supply cool down. (And while you're waiting, you might also want to fix whatever caused the short circuit or overload in the first place!)

OEM power supplies

Pin cab builders often use a mix of power supplies that includes one or more generic, no-name power supplies from eBay that look similar to the ones shown at right. These are often sold on eBay as LED strip PSUs or OEM PSUs, since they're primarily designed for sale to other manufacturers to embed in consumer devices.



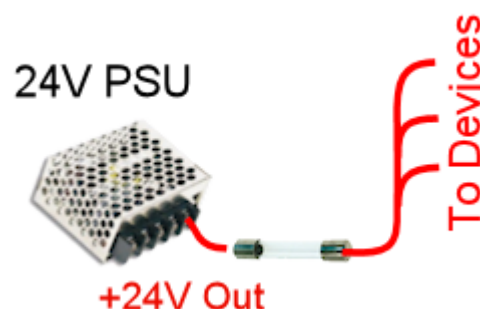
These power supplies might or might not have built-in overload protection. Check the site where you bought your to find out if they say anything about it. Also check to see if it has its own replaceable fuse accessible from the exterior of the case (this is rare).

If there's no built-in overload protection, you might want to protect the power supply with a fuse.

Choose a fuse that matches the rated maximum output amperage for the unit. If the unit's output limit is only stated in Watts, you can determine the maximum amperage by dividing Watts by Volts, using the voltage on the output side. For example, if your power supply has 12V output and a maximum power output of 48W, the maximum amperage output is $48W/12V = 4A$. So you'd choose a 4A fuse.

(Earlier, we talked about a 75% rule of thumb, where we use a fuse rated for 75% of the maximum for what we're protecting. That applies when we're protecting a transistor or IC chip. Power supplies shouldn't need this adjustment, since the components they use aren't as vulnerable to brief current surges.)

Connect a power supply fuse as shown in the diagram below. (We show a 24V supply as an example, but the principle is the same for any voltage.) Run a wire from the power supply's positive (+) output to one end of the fuse. Connect the other end of the fuse to whatever devices the power supply will be powering.



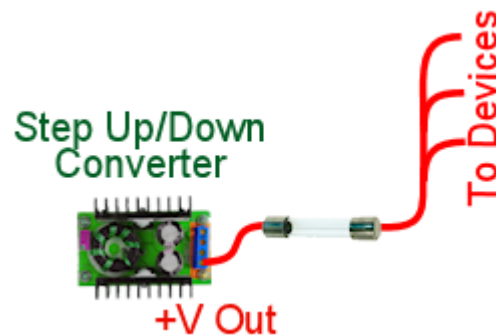
Step-up and step-down voltage converters

Pin cab builders sometimes use step-up and step-down voltage converters to get special voltage levels that you can't easily get from a PC power supply or eBay/OEM unit. For the purposes of fuses, you can treat these converters the same as power supplies.

Look at the instructions or spec sheet for the converter to determine its maximum output current. Select a fuse that matches the maximum current.

Note: If you're only using a converter to power a single device (e.g., a knocker coil or a shaker motor), and you already have a fuse in the circuit, you don't need a separate fuse for the converter. The first fuse will protect the whole circuit. Just make sure that its amperage limit is below the converter's maximum output amperage rating.

Wire the fuse for a converter the same way as with a power supply. Connect a wire from the converter's positive (+) voltage output to one end of the fuse. Connect the other end of the fuse to each device that the converter will be powering.



Fuses vs resettable devices

Fuses have the downside of being expendable: when a fuse blows, you have to throw it away and replace it.

It's possible to find resettable (non-expendable) circuit protection devices in the range of currents we use in a pin cab. I don't have experience with any of these devices myself, and I haven't found any options that look ideal for our purposes. But I wanted to mention them in case you don't like the idea of expendable fuses and wanted to look into reusable alternatives.

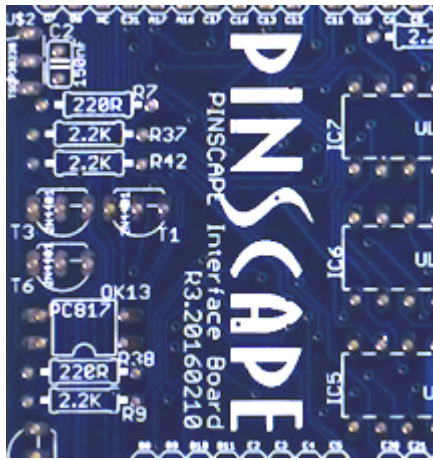
One thing you could look at is mechanical circuit breakers. These are similar to the ones you might find in the electrical panel in your house. There are options for these with suitable specs for a pin cab. The cheapest are a few times the price of an equivalent fuses, so they're more expensive if you never blow a fuse, but could end up being cheaper if the same circuit overloads several times.

Another possibility is PPTC (polymer positive temperature coefficient) devices. These are essentially temperature-dependent resistors that develop very high resistance when they get hot. High current levels heat them, triggering the rise in resistance, effectively cutting off current (or at least greatly reducing it). These devices have the advantages of being very compact, and being passive: you don't have to reset them after a fault (the way you do with a mechanical circuit breaker), since they return to normal resistance when they cool. PPTCs are widely used in PC equipment, including ATX power supplies.

The big problem I see with circuit breakers and PPTCs is that they're slow.

They tend to have timing curves similar to slow-blow fuses. That makes them good for fire prevention, but bad for protecting semiconductors - which is the primary function we need fuses for. For circuits where we need to protect transistors and ICs from short circuits and overloads, we need something that interrupts the current flow almost instantly on overload. The only thing I've found that does this is traditional, expendable fuses.

Part Five. *The Pinscape Controller*



85. Pinscape Controller Overview

This section leads you through the process of assembling the Pinscape expansion boards, setting up the KL25Z controller board, and setting up the software.

Here are the main steps involved:

- Buy a Freescale Freedom-KL25Z controller board. This is a retail product that sells for about \$15 to \$20 at any of the major electronics vendors. You can also find it on eBay and Amazon, but it's usually more expensive there. The KL25Z board is required for all systems, whether or not you're using the expansion board. It's included on the Chapter 91, electronics parts list, so you can order it along with any other components you need. This board comes fully assembled and ready to use.
- If you're going to build the Pinscape expansion boards, get hold of the blank printed circuit boards (PCBs) for the expansion boards. You can have a set fabricated by a PCB factory by sending them the freely available design files. You don't need any special electronics knowledge to do that; we'll explain the process. Alternatively, you might be able to get a set of pre-made boards from someone else in the pin cab community who made a batch for sharing.
- Buy the electronic components required for the expansion boards or any other external electronics you're going to include, such as a plunger sensor. Almost all of the components for the various options are fairly generic items that you can buy from any electronics vendor. We'll give you a complete list of parts with ordering links.
- Set up the Pinscape software on the KL25Z. You can do this before you build the expansion boards, plunger sensor, or other electronics, because the KL25Z can do some things on its own. If you're not in any hurry, you can also wait until after building any other
- Configure your KL25Z settings to match your hardware setup. This is done through a Windows program called the Pinscape Config Tool.
- Gather the necessary tools for building the expansion boards. We'll tell you what you need and provide buying recommendations for anything you don't already have at hand.
- Solder the components into the boards. You'll need basic soldering skills to do this, but you won't need any advanced electronics knowledge. We'll provide detailed instructions to make it as much of a "paint by numbers" job as we can.
- Connect the expansion boards to the KL25Z, to one another, and to your cabinet buttons, plunger sensor, and feedback devices.

What to buy

I hear from a lot of people who find themselves a little confused at this point about exactly what you have to buy, and what you have to build, to get this project going. The list of things to buy is outlined in broad strokes above, and in excruciating detail in the Chapter 91, electronic parts list, but that's a lot of material to go through. And there are several rather different ways to approach this project, which makes it even harder to sort out. So let's take a look at the most common scenarios, and spell out what's involved in terms of shopping.

Option 1: Standalone KL25Z (no expansion boards)

The KL25Z on its own can handle button inputs, accelerometer-based nudging, and it can connect to any of the plunger sensor options that the Pinscape software supports. It can also control feedback devices if you add separate external booster circuits for each device.

The shopping list for the standalone KL25Z is pretty simple: you really just need the KL25Z itself. The KL25Z is a retail product that comes fully assembled. You can find it at the electronics vendors like Mouser, as well as general retailers like Amazon and eBay. (A few people have asked if I sell these; I don't.)

One small qualification to "comes fully assembled": the KL25Z doesn't come with "pin headers" for plugging in cable connectors, because they wanted to leave your wiring options open. I recommend adding standard pin headers so that you can use it with standard plug-in connectors. This requires buying the pin header parts and soldering them to the board. It's an easy bit of soldering, and the required parts are listed under "KL25Z Microcontroller (standalone)" in Chapter 91, Electronic Parts List.

Option 2: DIY Expansion boards

The expansion boards give you all of the features of the standalone KL25Z version, and add an expandable set of output ports for feedback devices, plus built-in support for IR remote control signals to power on your TVs and monitors during system startup. The feedback device ports can support all of the common pin cab feedback devices without any additional circuitry, including motors, solenoids, contactors, and high-current LEDs.

There are three types of expansion boards (the main board that interfaces to the KL25Z, the "MOSFET power board" that can control high-power feedback devices like solenoids and motors, and the "chime board", for controlling certain types of pinball coils that have to be protected from long periods of continuous activation). The main board is always required, and the other two boards can be used in any combination (including two or more of each), according to how many output devices of each type that you plan to attach.

The shopping list for the DIY expansion boards includes the KL25Z, the blank circuit boards for your selected expansion boards, and a big list of electronic components (resistors, capacitors, transistors, IC chips, etc.) that have to be soldered into the blank boards.

The electronic components are all laid out in the Chapter 91, parts list. There's a section for each board, so you just go through those and include the parts listed for each board you're including in your system. You can get everything on the list from an electronics vendor like Mouser or DigiKey.

The blank circuit boards are custom-designed, so you have to get those made by a PCB maker. They're not something you can buy off-the-shelf from Mouser or Amazon. The plans for them are "open source", so anyone can have them made by any PCB maker of their choosing. There are three main ways to get them:

- Have a set made on demand by a PCB maker. There are lots of PCB companies that cater to hobbyists, with low prices and minimum order sizes of only 5 to 10 copies of each board. Most of the hobbyist vendors also manage to make the ordering process pretty easy. We provide step-by-step instructions in Chapter 93, Fabricating the Expansion Boards.
- Participate in a group order with other pin cab builders. Ask on the forums if anyone else is interested, and get together a few people to go in together on an order. This is a way to dispose of the extra copies you'd get in a minimum order from a PCB maker if you don't need all 5 to 10 copies for yourself.

- If you're in the US, you might be able to get them from me. I've been running a sort of ongoing "group order" for a couple of years now, but it's not the sort of group order most people run, where you have to wait for enough people to join the order in advance; instead, I've just been ordering small batches and sending them out to people on request. I'm not exactly "selling" them; I only charge what the boards cost me (plus shipping). I can't guarantee that I'll always have boards on hand or that I'll continue doing this indefinitely, so get in touch to get the current status. You can reach me on vpforums by private message; my user ID there is **mjr**. With apologies to those outside the United States, I'm afraid that I only ship to US addresses.

(Former) Option 3: Oak Micros boards

Note: Oak Micros announced in June 2021 that they're no longer shipping their boards, so the options below are not currently available. The discontinuation sounds permanent, but I'm going to leave the information below in place for now, if for no other reason than historical reference. I'm sure that the forum threads listed below will be updated if Oak Micros ever resumes selling the boards, so check the threads for current status.

Up until June 2021, Oak Micros offered fully assembled and tested versions of the Pinscape boards. The following options were available:

- Pinscape All-in-One board. This combines all of the functions of the DIY version's main board, power board, and chime board into a single circuit board, which makes it a little more compact and tidier to install than the DIY version. See this thread on vpforums.org for full details: Announcement: Pinscape All-in-One product.
- Pinscape Lite board. This is a less expensive board that includes a subset of the DIY board features: 24 input buttons, plunger sensor input, 12 high-power outputs for solenoids and motors (2 with PWM, 10 without), 16 low-power PWM outputs for LEDs, and the standard built-in accelerometer (nudge) sensor. This board isn't expandable, so it's best for mini-cabs and more basic cabs where you only plan to install a limited set of feedback toys. Details: Announcement: Oak Micros Pinscape Lite.

Plunger sensors

Regardless of which option you're going with for the Pinscape controller itself, the plunger is a separate matter with a separate parts list. The Chapter 91, electronic parts list includes separate sections with the parts needed for each plunger sensor design.

86. Expansion Boards Overview

The Pinscape expansion boards are a set of printed circuit boards that you can build yourself to add capabilities to the basic Pinscape system that runs on the KL25Z. The main functions of the expansion boards are:

- An almost unlimited number of high-power outputs for feedback devices (lights, solenoids, motors, etc) with full PWM brightness/speed/intensity control
- Circuitry for switching on a TV at system startup, for TVs that don't automatically turn on when the power is connected
- Neater wiring connections for buttons and plunger inputs

Expansion boards vs. Standalone KL25Z

The expansion boards are one way to run the Pinscape software. The other way is to run it directly on the KL25Z, without the expansion boards, which we refer to as the "standalone" KL25Z.

All by itself, the KL25Z can handle accelerometer-based nudging and button input, without any additional circuitry required. The only wiring involved is the wiring between the KL25Z and any buttons you want to connect. You can also connect any of the compatible plunger sensors directly to the KL25Z. The plunger sensor wiring is a little neater and more self-contained with the expansion boards, but there's no difference in how you build the sensors.

What the KL25Z *can't* do well on its own is controlling feedback devices, such as solenoids and lights. There are three big limitations on the bare KL25Z's capabilities in this area:

- It has a limited number of GPIO pins (the KL25Z's electrical connections to the outside world). Each output device you want to connect requires its own GPIO pin, but so does each button input. The plunger sensor and IR devices also require one or more pins each, if you want to include those in your system. The KL25Z has a total of about 40 pins that you need to divide up among these various functions, so the more buttons you want to include, the fewer feedback devices you can have, and vice versa. The KL25Z by itself simply doesn't have enough pins for all of the buttons and feedback devices in a well-equipped pin cab.
- Separately from the limited number of GPIO pins, the KL25Z only has 10 PWM channels. PWM ("pulse width modulation") is needed for LED outputs to control brightness, and for motors to control speed. 10 PWM channels isn't enough to cover a standard set of five RGB flashers (which require 15 channels: one per red/green/blue color channel per flasher).
- The GPIO pins can only control a tiny amount of voltage and current (no more than 3.3V and 4mA, which is barely enough for the smallest LEDs). External circuitry is required to boost this to a level that's sufficient to control a flasher LED, motor, or solenoid.

The biggest difference the expansion boards make is their extensive feedback device control capabilities. The expansion boards overcome the KL25Z's limitations on the number of available pins and PWM channels by adding an external set of PWM controller chips that provide nearly unlimited additional output ports, all with full PWM control. The expansion boards also overcome the KL25Z's power limitations by adding MOSFET boosters on the outputs. A MOSFET is a type of transistor that can control very high-power loads, so the MOSFET boosters make it possible to connect

all of the standard pin cab feedback devices directly.

The expansion boards also add circuitry for powering up one or more TVs at system startup. That's another bit of circuitry that you can build on your own (the circuit diagrams are included in this guide), but the expansion boards make it a little easier by integrating it into the board design.

DIY vs. Buy

Most of the rest of this section of the guide is about how to build the DIY design for the expansion boards. DIY means that you do the whole "manufacturing" process yourself. We provide CAD files for the circuit board designs, which specify the exact layouts and component lists for the boards. They're open-source designs, so you're free to modify them as you wish, or you can just upload them as-is to any PCB maker to have them made into physical boards. The designs include complete parts lists with the exact components needed, for ordering from an electronics supplier such as Mouser or Digikey. Once you have the blank PCBs and the components, you solder the components onto the boards. I know it sounds complex, but the individual steps are all pretty straightforward, and they're explained in detail in the pages ahead.

If you don't like the idea of building the boards yourself, though, there's at least one option¹ (as of this writing) for buying pre-built boards that come fully assembled and ready to use:

- L'atelier d'Arnoz (Arnoz's Pin Cab Shop). Arnoz sells a collection of circuit boards of his own design that work with the Pinscape software and replicate many of the functions of the DIY expansion boards. Like the original expansion boards, Arnoz's boards comprise a set of modules that you can combine to build your overall system. See his site for details and purchase options.

Which is better, DIY or pre-built? They both have their trade-offs. The big advantage of the DIY approach is that you can customize everything, from the hardware to the software, since the entire design is open-source. But it requires a lot of time and effort. Buying something ready-to-use is probably a better option if you don't have a hobby interest in building your own electronics. The price of the pre-built boards has so far been similar to what it would cost to build them yourself.

DIY: the three-board set

For do-it-yourselfers, the expansion boards are a system made up of three separate circuit board designs: the main board, the power board, and the chime board. The boards work together, and each board serves a different function. You don't have to build the full set; you can use the boards in different combinations to suit your needs. The main board is the only one that's required in all systems. The other two are optional, and you can add one or more of each according to the number and mix of feedback devices you want to connect.

Which boards do I need?

For most people, I recommend one main board and one power board. That covers all of the specialized functions like the TV control circuits, and gives you a total of 65 feedback device outputs: the main board's dedicated outputs for the RGB flashers, strobe, replay knocker, and flipper button LEDs, and the power board's 32 general-purpose outputs for everything else (button lights, contactors/solenoids, shaker motors, gear motors, fans, beacons, undercab LED strips). This is enough for a system that includes every feedback device described in Feedback Devices, and still leaves four or five spare power board outputs in case you can think of anything else

(and find room for it in your cab!).

You might also consider adding one or more chime boards if you're using any pinball coils or other solenoids that are vulnerable to rapid overheating. For pinball parts, this applies to almost any coil-based assemblies designed for the real machines: Chapter 64, chime units, bells, slingshot assemblies, or bumper assemblies. Flipper assemblies are sometimes safe, but not always; see Chapter 54, Coil Timers for details. If you do decide to add chime boards, each board will handle eight coils. Any chime boards will simply add more ports to your system, so you'll still have all of the ports on your main board and power board(s).

The chime board isn't strictly necessary even for coil devices, since the Pinscape firmware can emulate the timer protection via its "Flipper Logic" feature (also described in Chapter 54, Coil Timers). The main reason to use the chime board instead of software is that I consider a hardware timer to be inherently more reliable than a software version. That's not to say that the firmware isn't also reliable; it's just that software is inherently more complex, and can fail for reasons unrelated to the task at hand. The hardware timer is immune to software faults, so it's almost certain to do its job no matter what else goes wrong in the system.

Main board

The interface to the KL25Z. The KL25Z plugs into sockets on this board. This board is required for all expansion board systems, since it's where the KL25Z connects, and you can only use one of these (unlike the other boards, where you can use two or more if you need more ports).



This board provides:

- A pin header for up to 24 button inputs. This is just a convenience compared to the standalone KL25Z, where the GPIO pins for buttons are scattered around the headers. This groups all of the buttons into one connector.
- A pin header for connecting a plunger sensor. All of the Pinscape plunger designs in this guide use a common connector plug layout that fits this header. As with the buttons, this is just a convenience; on the standalone KL25Z, the pins for plunger connections are scattered around, but here they're neatly grouped into one connector that works with all of the plunger types.
- Outputs (grouped onto a single pin header) for a set of 5 RGB flashers, with high-resolution, high refresh rate PWM control, for connecting a Chapter 56, flasher panel (a feedback device consisting of five high-intensity RGB LEDs, usually placed at the back of the playfield TV). These outputs have sufficient power to supply two flasher panels in parallel, in case you want to install one both in the main cabinet and on the backbox.
- A dedicated output for a bright white strobe lamp, which is another popular feedback device typically deployed along with the flasher panel.
- Outputs (grouped onto a single pin header) for an additional 16 channels of low-power (20-50 mA) LEDs. This is aimed primarily at powering four small RGB LEDs for your flipper buttons (one for each flipper button and one for each MagnaSave button), which leaves four additional channels that can be used for other small LEDs, such as front panel button lamps.
- A dedicated output for a replay knocker. This has enough power handling for a pinball knocker coil, so you can wire it directly without any other circuitry (unlike, say, an LedWiz, where you'd need a relay or MOSFET booster for such

a high-power device). This output also has a special timer protection circuit that protects the knocker from software faults that could leave it stuck on, which is an occasional problem with PC pinball software that can destroy the coil by overheating it. The timer protection ensures this can never happen by shutting off the coil after a couple of seconds even if the software crashes. See Chapter 54, Coil Timers for more about the timer protection circuit.

- A "TV ON" relay that you can hard-wire to the soft on/off button on your TV, so that the software can turn on the TV at system startup. See Chapter 114, TV ON Switch.
- An IR transmitter, which can be used to transmit the remote control codes for your TV to turn them on when you power up the system. This is a less invasive alternative to the hard-wired on/off button connection. The IR transmitter can also be controlled from Windows, so you can make it send whatever IR codes you want at any time.
- Circuitry that detects when the system is powering up, so that the software can tell when it's time to send the remote control codes or switch signals to turn on your TVs.
- An IR receiver, which can be used to teach the Pinscape software the IR codes for your TV's remote control, so that the software can send the codes via the IR transmitter when it's time to turn the TV on. The IR receiver can also be programmed to send key presses to the PC when it receives certain codes, so you can use it as a way to access more control signals without adding physical buttons to your cabinet.

Power board

Adds 32 general-purpose high-power feedback device control ports, which can be used to control almost any sort of feedback toy. Each port can directly handle about 5 Amps and up to about 50V, which is enough for shaker motors, gear motors, fans, beacons, contactors, and solenoids. They'll also work just fine with LEDs and lamps, including fairly long lengths of LED strips (the non-addressable kind, anyway; you need a separate, dedicated controller for the addressable kind).



Every on the power board port has high-resolution, high refresh rate PWM control, so these ports provide brightness control for lighting devices, full color mixing for RGB devices, speed control for motors, and intensity control for solenoids.

The power board is an add-on to the main board. It's not required, but most people use one main board and one power board, since the main board doesn't have any general-purpose feedback outputs of its own (all of its outputs are for more specific purposes).

Multiple power boards can be daisy-chained. The Pinscape software can handle up to 128 feedback ports in total, which is enough for three of these boards if you're not using all of the main board ports. In practice, one power board is enough for a very decked-out system.

Chime board

This is another optional add-on to the main board, adding eight high-power outputs with timer protection circuits, for controlling chime units and pinball solenoids. These outputs have the same power handling capacity as the power board outputs, but add cut-off timers to prevent attached devices from being activated for more than a

couple of seconds at a time. These outputs are exactly like the dedicated replay knocker output on the main board.

See Chapter 54, Coil Timers for more about why timer protection is helpful for chime units and some other types of original pinball equipment.

Like the power board, the chime board can be daisy-chained, so you can add as many of these as you need, as long as your overall system doesn't exceed the 128-port limit of the Pinscape software. You can use any combination of chime boards and power boards that suits your system.



Schematics and board layouts

The boards were designed using a CAD program called EAGLE, from Autodesk. EAGLE works in terms of schematics and board layouts. A schematic is a symbolic, visual representation of the components in a circuit and how they're connected to one another. A board layout is the physical design of the circuit board, showing the locations of drill holes, solder pads, copper traces, and so on. The schematic and board layout are really two views of the same information.

You can download the EAGLE files for all of the expansion boards here:

mjrnet.org/pinscape/expansion_board/download.php

The downloads are ZIP files containing the following main file types:

- **.sch** files are the schematics for the boards
- **.brd** files are the physical circuit board layouts

The files also contain JPEG snapshots of the board layouts and PDF printouts of the schematics, in case you want to peruse the circuit plans without going to the trouble of installing EAGLE.

A free version of EAGLE (with some feature limits, naturally) is available if you want to view the plans interactively. See the Autodesk site linked above for downloads. Monthly subscription plans to the premium versions are also available. I recommend installing at least the free version if you're going to build the boards, so that you can explore the schematic and board layout files in detail - that's especially useful if you need to do any troubleshooting or debugging after building them. Installing EAGLE also gives you the ability to edit the plans if you want to customize them.

EAGLE is, unfortunately, rather difficult to learn. It has an extremely idiosyncratic user interface that goes against many of the Windows conventions. EAGLE somewhat makes up for its bizarre UI by being powerful and competent, once you get past the steep learning curve. To help with that, there are numerous tutorials and guides and videos available on the Web - search for "EAGLE tutorial". EAGLE is hugely popular with electronics hobbyists, so there's lots of help out there on getting started with it.

¹ In the past, there was another source for pre-built boards, Oak Micros, which sold a couple of boards based directly on the Pinscape expansion boards. Those are no longer available, but for historical reference, here's the original announcement: Pinscape All-in-One product.

87. Tools

You'll need a few tools to build the Pinscape projects. Most of these are the basics that you'd need for any electronics project. This section lists what you'll need and has some recommendations for the more specialized items.

Where to buy

Pololu is a great resource for specialized electronics tools. They hand-pick everything they sell specifically for its utility to robotics and Arduino hobbyists, which makes it much easier to find suitable items here than at a general retailer like Amazon, where you have to sort through a lot of less relevant offerings.

Amazon, has a much wider range of options than a specialized shop like Pololu, which is both a plus and a minus - a minus in that it's harder to find the items that are particularly for our kind of project.

Hardware stores and home stores like Home Depot and Lowe's are good sources for the basic hardware items like screwdrivers and pliers. They're not as good for electronics tools, since they don't carry many of those in the first place, and what they do carry is aimed at electricians doing house wiring rather than at computer-type electronics.

Miscellaneous workspace tools

Good things to have on hand in your work area:

- Magnifying glass
- Strong light

Screwdrivers

A good basic assortment of screwdrivers is pretty essential in everyday life, so you probably already have this one covered. You'll just need a range of sizes of Phillips and flat-head screwdrivers.

For electronics projects, the smaller sizes are most useful, particularly Philips #1 and #2. It's also helpful to have a set of small optician/eyeglass screwdrivers on hand.

Needle-nose pliers

Another toolbox basic. You don't need anything special here; a basic set from a hardware store should be fine.

Tweezers

Not essential, but handy to have when assembling circuit boards. Look for the anti-static (ESD safe) type - that will turn up options suitable for electronics work. I find the type with a curved or angled tip especially useful.

Wire cutters

This is another everyday toolbox basic, but the type in your toolbox might be more

suited for the large-gauge wire used in house wiring. You might want to pick up a smaller cutter more suited to the small-gauge (20-24 AWG) used for most Pinscape and pin cab wiring. Look for a "flush cut" type. Hakko makes a few "micro cutters" that work nicely.

Wire stripper

This is a tool that removes the insulation from a length of wire. This is another one that you might have in your toolbox for everyday jobs around the house, but as with wire cutters, the one in your toolbox might be mostly for larger-gauge house wiring. Look for a wire stripper that has slots specifically for wire gauges 16, 18, 20, 22, and 24.

"Dupont" cables

For testing purposes, it's extremely handy to have a collection of "Dupont" cables on hand. These are hookup wires with male and/or female 0.1" pin header connectors attached at each end. They're great for testing the expansion boards, because they make it easy to test a connection to any individual pin on any header. Most of the connectors on the expansion boards use the same 0.1" pins that the Dupont connectors are for.

You can find sets of these cables on eBay for extremely cheap, if you're willing to wait a month for shipping from China. The same cables are also available at all of the hobby robotics Web stores - Pololu, SparkFun, Adafruit, etc, and of course from Amazon. Search for "Dupont cable" at any of those sites. You'll also see them described as "breadboard jumper wires".

Pick up a full set with all of the gender combinations - male-to-male, female-to-female, and male-to-female.

Alligator clip cables

Also for testing purposes, you might want to pick up a set of cables with "alligator clips" at the ends. Alligator clips are just little spring-loaded metal clips that you can connect to wires, pins, and terminals of just about any kind, so they're great for making temporary, ad hoc connections when testing.

You can find these at all of the same places where the Dupont cables are sold.

Voltmeter/Multimeter

It's not strictly necessary as a build tool, but a multimeter is pretty essential for debugging. All of my EE friends swear by their Fluke meters, and I agree that they're probably the best. But the Fluke meters are really expensive, and my own experience is that the cheaper brands are just fine for hobbyist electronics like a pin cab project. You can find perfectly good meters for under \$20.

Look for a model that can measure the following:

- Volts DC
- Volts AC
- milliAmps/Amps DC
- Ohms (resistance)

- Continuity testing (preferably with an audible signal)

The cheapest models have "manual ranging", meaning that you have to set a dial to a range of values that you want to measure. For example, the dial might have a series of DC voltage settings with labels like 100mV, 1000mV, 10V, 100V, etc. This means that you have to set the dial to the range you expect to measure. In contrast, an auto-ranging meter has a single dial setting for "Volts DC", and automatically figures out the right range to use to display each reading.

I'd recommend going with an auto-ranging meter, since they're easier to use, and they're not much more expensive than the manual type.

Soldering tools

You'll need a soldering tool of some kind for just about any pin cab build. If you're building the Pinscape expansion boards or any other circuit boards, you'll obviously be doing some soldering work. But even if you're not building any circuit boards, you'll probably run into at least a few places in your cab where you'll need to solder a couple of wires together or solder a wire to a terminal.

There are two very different types of soldering tools: soldering irons and soldering stations. A soldering iron is a cheap tool (about \$10) for household jobs; a soldering station is a more expensive professional tool (\$100 and up) for electronics work.

A soldering iron is fine if you're not building any circuit boards; it'll be adequate for a few miscellaneous connections in a cab. If you're planning to do any circuit board assembly, however, I strongly recommend investing in a soldering station. This is one of those cases where the right tool makes the job much easier, and makes your work product much better.

The big difference between a soldering station and an iron is that a station has a thermostat that precisely maintains a temperature on the tip, and has an adjustable temperature setting. This alone makes a world of difference, since having the tip at a consistent temperature makes the solder behave predictably. Soldering stations also heat up much more quickly than irons, typically in 30-60 seconds, and they tell you when they're ready, so there's less waiting around and no guesswork. The stations also tend to use much higher quality tips than basic soldering irons do, and the tips can be changed for different types of work.

There are two models in the \$100 range that seem to be the "standards" among hobbyists: the **Hakko FX888D** and the **Weller WE1010NA**. Both have similar features and are widely used. I use the Hakko and have been quite happy with it.

Desoldering pump ("solder sucker")

This is a tool to *remove* solder, so that you can remove a previously soldered part (to replace a dead part, for example) or just remove excess solder if you apply too much.

This is more of a nice-to-have than a must, but it can come in handy.

The type I've had the best luck with is what's known as a manual de-soldering pump or "solder sucker". This is a fairly cheap tool, but pretty effective. You can find these on Amazon for \$10 to \$25.

The other cheap option is "solder wick", which is supposed to soak up solder, but I've never had very good luck with this.

There are also dedicated de-soldering stations, with powered solder pumps. These

are in the same price class as soldering stations, so I wouldn't buy one for occasional hobby electronics. But I'd definitely consider one if I did a lot of repair work, since the manual tools require a great deal of patience when they work at all. The powered de-soldering stations are supposed to be vastly better, as power tools usually are.

Solder

It might seem like "solder is solder", but some types of solder actually work much better than others. This is another case where the right tool will give you better results regardless of skill level.

The reason for the variability is that solder is actually a pretty complicated formulation. It's a mixture of metal alloys and "flux", which is a chemical that affects how the solder behaves when melted. The mixture of metals controls the melting point of the solder, and the flux controls how the molten solder flows, coheres, and adheres to wires and surfaces. Different formulations are appropriate for different jobs; the type you'll find at Home Depot really doesn't work all that well for fine electronics work.

My top pick is **Kester 44 Rosin Core 63/37** solder in the .031" size. This is an all-around great solder for circuit board work.

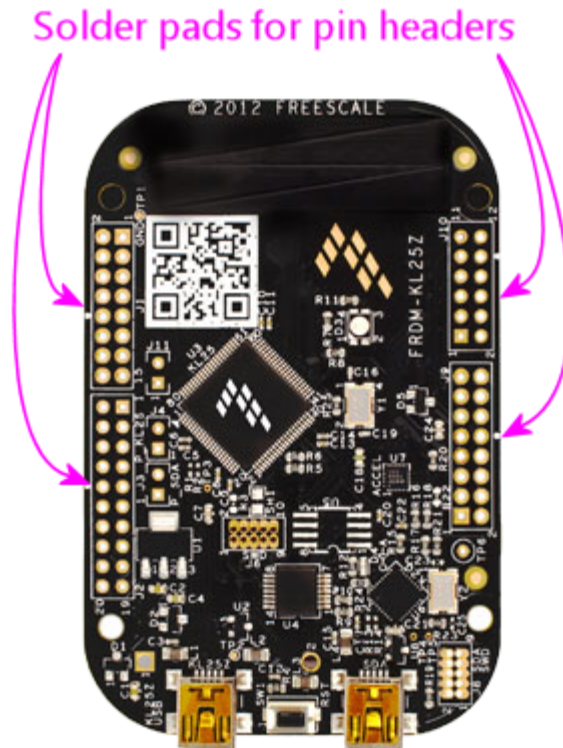
Crimp tool

If you're planning to assemble any hand-made connectors using crimp-pin housings, a crimping tool is essential. See Chapter 82, Crimp Pins for tool recommendations.

88. KL25Z Hardware Setup

The KL25Z comes fully assembled and ready to use. All you have to do to get it working is plug it into your computer with a USB cable.

If you're going to connect any additional hardware to the KL25Z (plunger, buttons, feedback devices), there is one bit of additional hardware assembly you'll have to do. You'll need to install "pins" for plugging in connectors to the other devices. They don't install these at the factory; they just include empty solder pads where you can install your own pin headers if you wish to.



The KL25Z will run happily without the headers installed, so you don't have to do this right now. You can skip ahead to the Chapter 89, software setup, and come back here later, if you want to try out the board first.

The specific headers to buy (along with the mating connectors, known as "crimp pin housings") are listed in the "KL25Z Microcontroller (Standalone)" section in Chapter 91, Electronic Parts List. They look like this on their own:

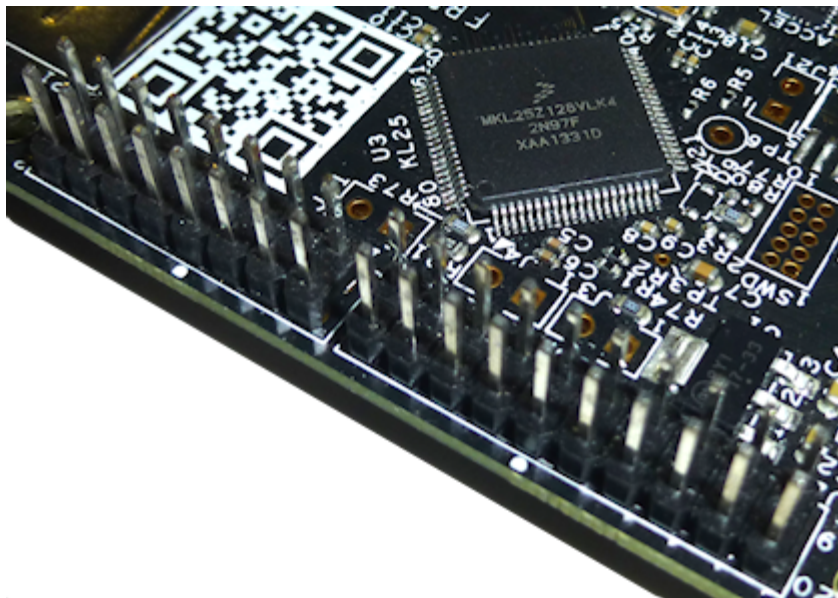
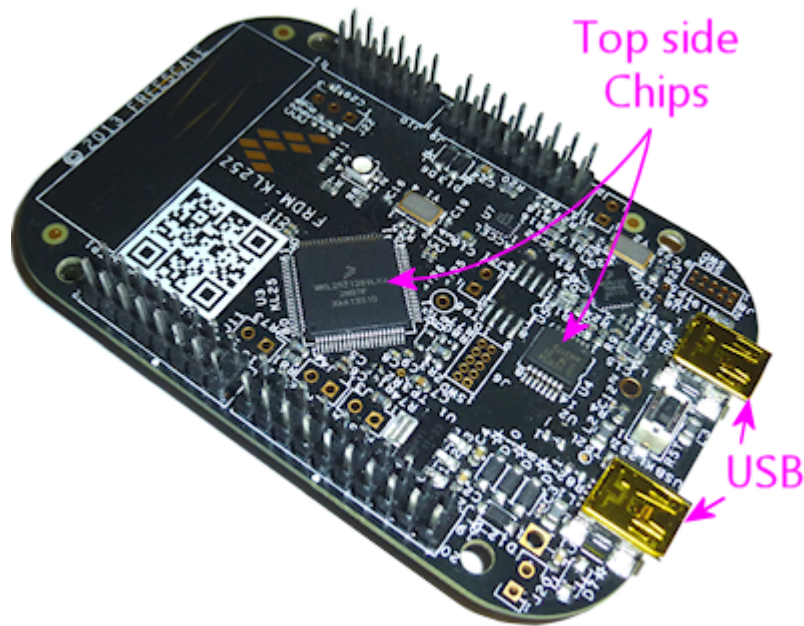


The installation procedure is a little different depending on whether you're using the KL25Z "standalone" or with the Pinscape expansion boards. Continue to the appropriate section below.

Standalone KL25Z pin headers

If you're using the KL25Z **without** the Pinscape expansion boards, the pin headers go on the top side of the KL25Z - the side with the large CPU chip and the USB

connectors. Here's how they look when installed:



Important! Install the pins on the top as shown **only for standalone use**, not for the expansion boards. The expansion boards require the pins to be installed on the bottom side of the board instead.

The pin headers are pretty easy to install:

- Fit the headers onto the solder pads with the **shorter** pins facing the board
- Seat the plastic base flat against the surface of board
- Solder the pins on the bottom side of the board

Be sure to solder **all** of the pins. Some people ask if it's okay to just solder a couple of the pins, since that makes a strong enough mechanical connection. It's not; you really do have to solder all of them. The solder is there for the electrical connection between pins and pads, so each pin needs its own solder joint. See Chapter 81, 0.1" Pin Headers for details.

The mating connectors in the parts list are "crimp pin housings". See Chapter 82, Crimp Pins for help assembling them. These are ideal for wiring to the KL25Z

because they make it easy to connect each pin individually to a separate piece of hookup wire. Each pin has its own unique function - some pins are for button inputs, some are for connecting a plunger sensor, some are for feedback outputs. The individual wiring per pin makes them take a little work to assemble, but it pays off in the long run because it lets you customize the wiring exactly to fit your cabinet.

Warning on KL25Z GPIO pins

Never connect any output devices directly to the KL25Z. The KL25Z GPIO pins have extremely low power handling limits:

- Maximum 3.3V
- Maximum 4mA

Exceeding these limits can destroy the entire KL25Z, because the GPIO pins are wired directly into CPU core. Applying higher voltage or current levels can overheat the CPU chip and destroy the whole thing.

You *can* directly wire button inputs to the KL25Z, as long as you follow our wiring plans in Chapter 110, Pinscape Button Inputs. You can also connect any of the plunger sensor designed described in this guide, as long as you follow our wiring plans.

The main thing that you should never connect directly is any sort of feedback device - flasher LEDs, LED light strips, strobes, button lamps, motors, solenoids, contactors, relays, etc. You always need some sort of "booster" circuit with feedback devices. We provide several options in Chapter 49, Pinscape Outputs Setup (Standalone KL25Z).

How to approach wiring with the standalone KL25Z

You'll notice that the pins on the KL25Z are arranged into four groups, for four separate connectors, so you might think it would be nice to use these to group pins by function: one connector might have all of the button inputs, say, and another might have all of the feedback outputs. Believe me, that occurred to me as well. Unfortunately, it really can't be done. The problem is that each pin on the KL25Z has its own special-purpose capabilities, which means that we can only use certain pins for certain functions. And these capabilities aren't grouped very neatly in the KL25Z pin layout. That's not something we can change through software, since a lot of this is baked into the circuit design inside the KL25Z CPU. So the Pinscape functions have to be scattered among the headers.

That's where the crimp pin headers are handy. They let you connect each pin to its own hookup wire, so it doesn't really matter that the functions aren't grouped.

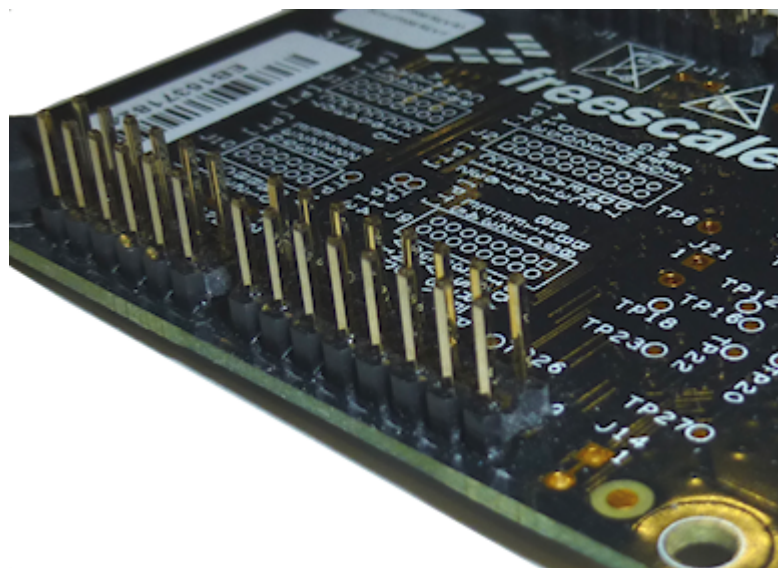
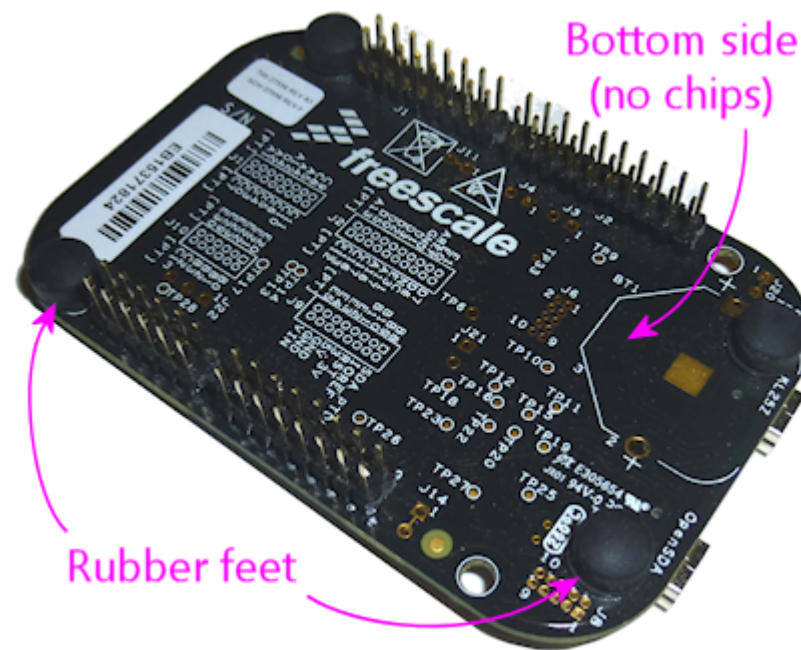
Here's how I'd approach the KL25Z wiring:

- Don't worry about pre-assembling any of the housings. You can leave any or all of the pin slots empty in a housing, and you can insert one crimp pin at a time, so you can fill in the pin slots as you need them. Start with all of the connectors empty. When you install a button in your cab, add the wiring for the button with a hookup wire leading back to the KL25Z, crimp a pin on the end of the wire, and insert it in the housing in the appropriate slot.
- Treat the wires as permanently installed in the cabinet, but treat the things connected to the wire as removable, so use pluggable connectors at the ends of a wire. For the KL25Z, the crimp pin connectors serve. At the other end:
 - For buttons, use "quick connect" push-on terminals, if possible. The SuzoHapp buttons use 1/4" (6.35mm) quick-connect terminals, compatible with 1/4" female spade connectors.

- For connections to other circuit board (such as a MOSFET booster for feedback outputs), use another similar pin header and crimp pin wire housing on that end.
- For a plunger sensor, I recommend building a Chapter 100, plunger sensor breakout board. That will give you an easy way to plug in any of our plunger sensor types using the standard plugs used on the expansion board. You can alternatively hard-wire the sensor to the KL25Z headers, but that breaks my rule about making everything easily unpluggable, and it'll make things harder if you ever need to remove the sensor for any reason.

KL25Z pin headers for the expansion boards

For use with the Pinscape expansion boards, the pin headers go on the **bottom** side of the KL25Z. The bottom is the side with the rubber feet, and **without** any chips. The chips and USB connectors are all on the top side.



To install the pin headers:

- Fit them onto the solder pads with the **shorter** pins facing the board

- seat the plastic base flat against the surface of board
- and solder the pins on the opposite side of the board (in this case, since we're installing the pins on the bottom side, the solder goes on the top side)

See Chapter 81, 0.1" Pin Headers for more about the pin headers in general.

89. KL25Z Software Setup

The KL25Z is a "microcontroller", which is basically a tiny computer. Unlike a desktop computer, though, it doesn't require an operating system in the usual sense. There's no equivalent of Windows or Linux to install. Instead, you only have to install one thing: the Pinscape firmware. That serves as both the operating system and the application software. It controls all of the virtual pinball functions of the KL25Z, including the sensors, buttons, and feedback devices, and it handles communications with Windows.

In addition to the Pinscape firmware that runs on the KL25Z, there's a separate Pinscape program that runs on your Windows PC, called the Config Tool. This provides an interactive interface for setting up the device, configuring it, and testing it. You don't need to leave the Config Tool running all the time; it's only needed to set up the device. You can also run it again at any time to change options, update the firmware, or troubleshoot problems (it includes some testing features that can help debug the hardware setup).

The software installation process is all controlled from the PC. The Config Tool handles the KL25Z software setup, so the first step is to install the Config Tool on your PC. You can download it here:

PinscapeConfigTool.zip

To install, download the ZIP file above and unpack it into a folder on your hard disk. Use any location that's convenient. Open the folder and double-click the "PinscapeConfigTool" application.

The config tool should automatically go out and find the latest firmware version and download it for you. The firmware files are fairly small (in the 100K byte range), so this should only take a few seconds if you have a broadband Internet connection.

To install the firmware, click the link "Set up a new KL25Z" in the config tool window. The program will lead you through the process of setting up the device and installing the firmware.

If the Config Tool doesn't recognize your device at all, you might need to manually install a new version of the KL25Z boot loader; see **KL25Z Boot Loader update** below.

If all goes well, you should see a Pinscape device listed when you get back to the main menu screen. This will give you options to configure settings and test the device's inputs.

You'll also see the LED on the KL25Z flash a status indicator pattern. Alternating green/yellow or green/blue indicate healthy operation. See Chapter 90, KL25Z Status Lights for the full list of status light patterns.

KL25Z Boot Loader update

In the old days before about 2018, KL25Z's shipped from the factory with a "boot loader" that didn't work with Windows 8 or 10. The boot loader is the part of the device's software that lets you install new software - such as Pinscape - onto the device, so it's a pretty critical component. More recent versions of the KL25Z have a newer version of the boot loader that does work properly with Windows 8 and 10, so if you bought your KL25Z any time after 2018 or so, you probably won't have to worry about this. But just in case you happen to have found a vendor selling dusty old stock from the back of the warehouse, I wanted to mention this.

In most cases, the Pinscape Config Tool will automatically test the boot loader version when you use the **Set up a new KL25Z** procedure described above, and walk you through the update process if it's necessary. If you were able to go through that procedure successfully, you can skip the rest of this section. But if the Config Tool didn't recognize your device when you tried to set it up, you might need to refresh the boot loader manually. The procedure is below.

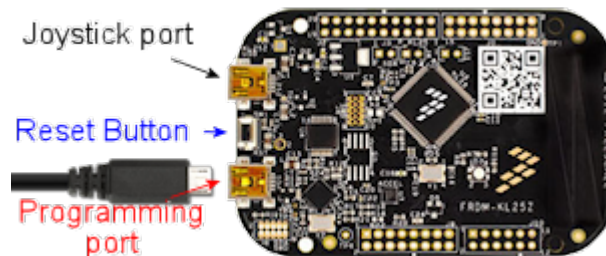
However, there's an important and rather obnoxious catch-22 you need to know about first: **This procedure might only work on Windows versions 7 and earlier.** It might not work on Windows 8 or 10. Old versions of the factory firmware had a bug that made them incompatible with 8 and 10. This is the obnoxious catch-22: you need the boot loader to *update* the boot loader, so if you have the old version that doesn't work on your Win 8/10 machine, you can't use it to install the new version that does work on Win 8/10. There are some complex workarounds that people have found, such as running Windows 7 in a virtual machine box. You can find that information in a Web search, but I'm not going to reproduce it here, because it's very complex and tedious, and no one should need it any more. If you just bought a KL25Z, and find that it doesn't work with your Windows 8/10 machine (and you don't have an old Win 7 machine lying around), I'm going to suggest that you send back the KL25Z as defective, and ask the seller to send you a fresher one that was manufactured in 2019 or later.

And again, if the Config Tool recognized your device and you were able to go through the **Set up a new KL25Z** procedure successfully, you **don't** need to worry about this procedure.

Finally, this is a **one-time procedure** that you should never have to do again for this particular KL25Z. It's not necessary to repeat this when updating Pinscape versions.

Here's the procedure:

- In your Web browser, go to pemicro.com/opensda/
- Find the section **OpenSDA Firmware (MSD & Debug)**. Click on the **Firmware Apps** link. You'll be asked to create a free account or to provide your email address to receive a download link. Choose the option you prefer and download the file.
- Unzip the downloaded file to a local folder on your hard disk.
- **Unplug all USB cables from your KL25Z.** If you have any other KL25Z cards already installed, unplug all of those as well.
- Open the My Computer window on your Windows desktop, so that you can view attached disk drives.
- **Press and hold** the KL25Z reset button.
- **Keep holding the reset button** while you plug a USB cable into the **programming** port on the KL25Z, and plug the other end into a USB port on your computer.
- Release the reset button.
- You should see a new thumb drive icon appear in the My Computer window, with a name like **BOOTLOADERAPP**. This represents the KL25Z's programming port. (The name might be different depending on the old boot



loader version. As long as a new drive appeared when you plugged in the USB port, that should be the right one, no matter what it's called.)

- In the PEmicro software you downloaded above, find the file **BOOTUPDATEAPP_Pemicro_v111.SDA**. (The digits at the end are the version number, so they can vary.)
- Drag and drop that file onto the KL25Z thumb drive.
- Wait for Windows to finish copying the file, then unplug the KL25Z.
- We're now going to repeat the steps above with a different .SDA file. As before, press and hold the reset button on the KL25Z, and plug in a USB cable between the KL25Z programming port and your computer. Release the reset button. In the downloaded files, find **MSD-DEBUG-FRDM-KL25Z_Pemicro_v114.SDA**, and drag it onto the thumb drive. (As before, the digits at the end might vary.) Unplug the USB cable after Windows finishes copying the file.
- To confirm that everything worked, reconnect the KL25Z programming port to your PC **without pressing the reset button**. The thumb drive should appear again, but it should now be titled **FRDM-KL25Z**. You should also see a steady (non-blinking) green LED on the card while it's plugged in.
- If any of this doesn't match what you see, the update might not have worked. **If you attempted the upgrade from Windows 8 or higher, that's probably the problem.** The old (pre-2018) factory firmware is incompatible with Windows 8-10 and will fail silently and mysteriously, with no error messages, if you're on Win 8-10. If you did the update from Windows 7 or earlier and it still didn't work, it's possible that the procedures have changed since this writing. Try looking for instructions (.txt and/or .pdf files) in the files you downloaded from PEmicro, to see if there's any information on new procedures.
- If all went as expected, your KL25Z is now ready to use. With the updated firmware in place, the device should now work on all versions of Windows from XP to 10 and hopefully beyond.

Software updates

I update the Pinscape firmware periodically to fix bugs and add new features. I always try to maintain compatibility with existing installations when making changes, so you can generally update to the latest no matter which version you started with. However, it's completely up to you whether or not you want to keep up to date with the latest changes. There's something to be said for not messing with a working system, but it's also nice to have the latest bug fixes and features.

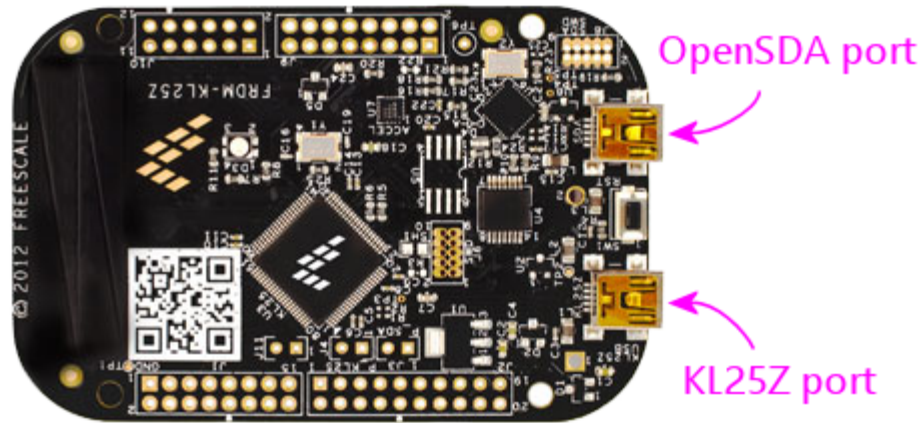
The Config Tool automatically checks for updates (unless you tell it not to) each time you run it, but it doesn't install them on its own. You have to tell it to install updates. So your firmware will never mysteriously change when you're not expecting it.

You can download the latest versions and archived versions of the firmware and Config Tool at the Pinscape web site:

<http://mjrnet.org/pinscape/swversions.php>

KL25Z USB ports

The KL25Z features two USB ports. They're not interchangeable - each port has a specific function.



You don't have to memorize which port is which. The ports are labeled in tiny text on the bottom side of the board. Just flip the board over and look for the labels if you're ever in doubt.

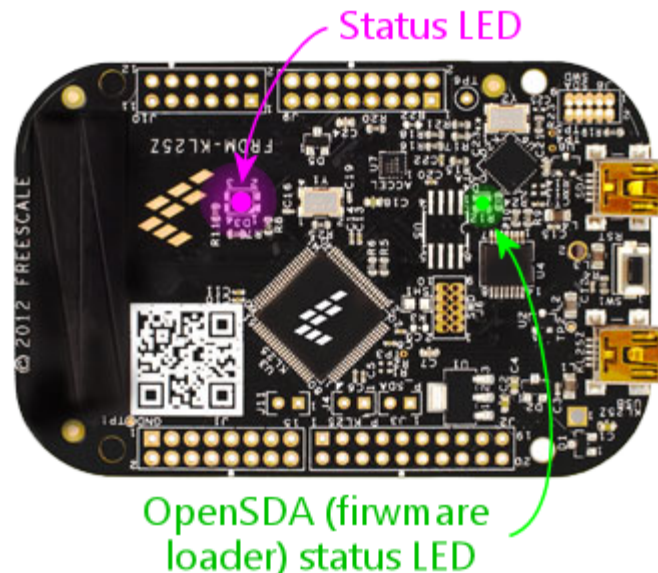
The "KL25Z port" is the one that Pinscape uses for all of its joystick emulation, keyboard emulation, and LedWiz emulation functions. This port must always be plugged in during normal operation.

The "OpenSDA port" is the "programming" port. It doesn't actually connect to the microcontroller proper; it's connected to a separate processor on the board that's just there to handle firmware downloads. This port is only needed when you want to update the firmware, so you don't have to leave it plugged in all the time. But you can; it doesn't interfere with normal operations. I find it more convenient to leave it plugged in all the time, especially in a pin cab setting where you'd have to open up the cabinet to plug it in manually.

The OpenSDA port emulates a USB thumb drive, so you'll see a virtual hard disk (usually called "KL25Z") appear on your Windows desktop whenever the cable to this port is plugged in. This isn't a true disk drive, so don't copy random files there - doing so can mess up the Pinscape firmware. This virtual drive's only function is to program the KL25Z's flash memory with new firmware.

90. KL25Z Status Lights

The Pinscape firmware blinks the LED on the KL25Z to indicate its system status. This is a quick way to check if the Pinscape software is operating normally (or, at least, to check if the software *thinks* it's operating normally, which might not always be the same thing!). If you suspect that something's wrong, or you're not seeing expected input on the PC, the LED lets you check at a glance what the Pinscape software thinks about its current health.



Here are the various flashing patterns you might see and what they mean:

- **Long blue/green** lights, alternating about every second: normal healthy operation.
- **Long yellow/green** lights, alternating about every second: normal healthy operation. The device has a good USB connection and is running normally. However, the plunger hasn't been calibrated yet, so you should run the plunger calibration process to finish device setup.
- **Short yellow flashes**, every couple of seconds: the device has just restarted and is trying to connect to the PC via USB.
- **Two short red flashes** in rapid succession every couple of seconds: the USB connection to the PC was broken. This can happen if the USB cable is physically unplugged *or* the connection is broken at a software level. Software disconnections can happen due to software errors, and can also happen if the Windows USB drivers intentionally disconnect.
- **Long red/yellow** lights, alternating: USB data transmission problems are occurring. The device still has a connection to the PC (or so it appears to the device), but the PC isn't acknowledging transmissions. The device will try to restore the connection automatically when this occurs, so this condition is usually short-lived; the device will usually either return to normal within a few seconds or will reboot itself to try to reset the connection.
- **Medium blue flash**, about half a second each: the TV ON delay timer is running. This means that the power to the secondary PSU has just been turned on, and the TV ON timer is waiting for the configured delay time before turning on the TVs. You'll only see this if the TV ON feature is enabled.
- **Fast red/purple**, alternating: the device is out of memory. This means that the programmed configuration is too complex. The KL25Z has a limited amount

of memory, and each feature enabled in the configuration consumes some of that. It's possible in principle to exceed available memory by enabling too many features simultaneously. This error should only occur *immediately after a reboot*, since the software allocates all memory up-front during startup and uses a fixed memory configuration from that point on. So you'll never see this happen spontaneously because of something you're doing during a pinball game session, for example. And at the moment, it really shouldn't be possible to trigger this condition at all with any real-world configuration. But you can do it by deliberately maxing out every possible configuration option. If you do happen to trigger this error, it will keep happening again every time you reboot, since it's a function of the saved configuration. The only way to clear it is to reset to the default configuration by reinstalling a fresh copy of the Pinscape software.

Re-purposing the LED for feedback output

The status indicator LED on the KL25Z is connected to GPIO ports (three of them - one each for the LED's red, green, and blue channel), which means that it's under the control of the Pinscape software program. That much is probably obvious given that the software uses the LED to display the status light patterns listed above. But it also means that you can assign the LED elements as feedback device outputs. The Config tool lets you assign the GPIO pins associated with the LED to DOF output ports, exactly like any other GPIO pins, in the Outputs list on the Setting page.

If you do assign an LED element as a DOF-controlled output port, the Pinscape software will cede the LED port to your control, and won't try to use it as a status light. That means that the flash patterns listed above won't appear if you take over the LED for output port use. Obviously, the LED can't serve two masters, so the Pinscape software gives precedence to your settings.

Why would you want to use the LED as a feedback device? The main use case I see is testing. If you want to test a new DOF configuration on your PC, for example, and you haven't set up any external feedback devices yet, the on-board LED could be used for a quick test that DOF signals are getting through to the KL25Z.

Finally, I'll point out a couple of odd quirks to the way the LED is wired on the KL25Z.

The first quirk is that the LED channels are wired in "active low" fashion, meaning that each LED color turns **on** when the voltage on its GPIO pin is **off**. When setting up the LED elements as outputs in the Config Tool, just be sure to enable the "Active Low" option for each port. If the LED shows the opposite of what you want (the LED is on when you want it off, off when it should be on), you probably just need to go set the Active Low option.

The second quirk is way the LED connections are wired. The red and green LED elements are wired to GPIO ports (PTB18 and PTB19, respectively), but this is all purely internal within the KL25Z circuit board: they're not wired to any external pins. Strangely, though, the blue LED element (PTD1) *is* wired to an external pin. So you can use port PTD1 to control something wired externally in addition to controlling the blue LED element. I don't recommend doing that, since it's pretty confusing to think about, but if you were in a pinch and absolutely needed one more external GPIO pin for some external control circuit, it would work. The blue LED would just be sort of dragged along with whatever you were doing with the external port, so you'd have to put up with the blue LED turning on and off in concert with the external circuit.

Using PTD1 for button input

The blue segment of the status LED is wired to GPIO port PTD1, which is also wired to one of the pins on the KL25Z's main pin headers (J2 pin 12; see Appendix 1, KL25Z Pin Out). The pin header wiring means that you can use PTD1 for other purposes besides controlling the blue LED - but if you do, it requires some special handling, because the blue LED connection is hard-wired on the board and can't be changed in software.

(The red and green segments are also wired to GPIO ports internally, but they don't have any pin header wiring, so there's no way to use those ports for anything other than the LED connections.)

You can use PTD1 as a feedback device output port simply by assigning it in the config tool. Assigning it as a feedback output will prevent the Pinscape firmware from using it as part of the status indication. However, it doesn't change the internal hard-wiring between PTD1 and the blue LED, so the blue LED will now turn on and off in sync with any external feedback device you wire there. (Actually, in sync, but opposite: remember that the LED uses "active low" wiring, so the blue LED will turn ON when the output port is OFF, and vice versa.)

You can also use PTD1 as a button input, again, simply by assigning it as a button input in the Config Tool. With this use, the blue LED segment will stop acting as part of the Pinscape status indicator; instead, it'll just light up whenever you press the button.

If you do use the PTD1 header pin for one of these uses, and it bothers you that the blue LED is affected by the port status, you can physically modify the KL25Z to sever the connection to the LED. The procedure is described in Appendix 1, KL25Z Pin Out. Be aware that severing the blue LED connection will change the Pinscape status light patterns accordingly, since the Pinscape firmware won't be able to light up the blue LED any longer.

OpenSDA status LED

There's a second status LED on the KL25Z: a smaller monochrome green LED, close to the USB connectors. This one shows the OpenSDA (firmware loader) status:

- Solid green means "OK" - the OpenSDA port is connected and the boot loader is awaiting commands
- Slow flashing means that a firmware update is in progress
- Rapid flashing means that an error has occurred

If rapid flashing appears, indicating an error, you can clear it by unplugging both USB cables, waiting a couple of seconds, and then plugging the cables back in. That should reset the device and return the status indicator to solid green. Once an error condition appears, it stays there until you clear it manually.

If you see an error indicator, it's not due to anything you did wrong. It's just random bugs in the KL25Z boot loader, as far as I can tell. That part of the software comes from the KL25Z's manufacturer, so it's not something I can easily fix. An error condition on the OpenSDA side won't affect normal Pinscape operations, so you can ignore the little green LED most of the time. It only matters when you're trying to update the Pinscape firmware.

91. Electronic Parts List

This section lists all of the electronics needed for each Pinscape Controller subsystem.

The ID column lists the "reference designator" for each component as it appears in the schematics. That's just an arbitrary ID assigned for cross-referencing between the schematic and the parts list. Each ID is unique within its board, but has no meaning outside of that. These same IDs are also printed on the circuit boards next to the parts, so that you can match up the physical components on the boards to schematics and the parts list.

Note that the parts lists don't include any of the custom PCBs (printed circuit boards), such as the expansion boards. These have to be custom fabricated, which is of course a different process than ordering off-the-shelf parts. We explain how to do get these made in Chapter 93, Fabricating the Expansion Boards.

Optional elements

The parts lists below reflect the "default" configurations for the expansion boards. There are some variations possible in how you build the boards, though, including some things you can just leave out if you don't need them. The list itself has a lot of footnotes that describe alternate parts or mention when something is optional, so you pay attention to the notes while building out your shopping cart.

In addition, you might want to read through "Optional elements" in Chapter 94, Building the Expansion Boards. That section explains how you can omit some of the features entirely, which for the most part is a simple matter of not installing the parts involved in the feature. That can save you a little time and effort for functions you don't need, and you can save on the parts cost by omitting the unnecessary items from your shopping cart.

Warning on TLC5940NT

The TLC5940NT chip, which is central to the design of the Main Board and Power Board, is no longer in production. That means that you can't buy it from mainstream electronics suppliers like Mouser and DigiKey. However, the chip *is* still available from sellers on eBay and Aliexpress. There seems to be an ongoing bottomless supply of the chip on those venues, which is both good news and bad news. The good news is that it means we can still build these boards, and apparently will be able to do so for the foreseeable future. The bad news is that the only way these chips can still be coming to market after being discontinued for so long is that someone is making unauthorized gray-market knockoffs. Chip counterfeiters aren't known for holding themselves to the highest quality standards. Indeed, reports on the forums suggest that the Dead-On-Arrival rate for these chips is quite high lately. There's not much you can do about it other than take your chances and hope you get a good batch. The only consolation is that the chips are pretty cheap, so the financial loss isn't huge if you do get a bad batch. But the cost in your time for the soldering and desoldering could be huge if you don't use sockets - please do use sockets for these chips so that it's easy to swap them out if necessary.

If you use the Shopping List tool below to generate an order list to upload to Mouser, the tool will show a warning that the list doesn't include the TLC5940NT, to remind you that you'll have to order those separately.

I would have redesigned the boards around an in-production substitute chip a long

time ago if such a thing existed, but alas, there simply are no similar chips available. Every similar chip that's currently in production is in what's known as SMD (surface-mount device) packaging, which is designed for robotic assembly and is difficult to solder by hand. I deliberately designed entirely around parts that can be soldered by hand, since one of the main goals of these boards is that you can build them yourself. So we're stuck with the gray-market TLC5940NT.

Substitutions

The lists below provide specific manufacturer part numbers for all components, with Mouser.com links to the parts.

These are **reference** parts only, not requirements. It's perfectly fine to buy the exact parts listed, but it's also perfectly fine to substitute equivalent parts wherever you wish. The main reason we list specific parts is to save you time shopping. There are so many options available for some of the parts that it can take quite a while to narrow the selection based on the specs alone.

If you have any trouble finding the specific part numbers listed, or they're out of stock at your preferred vendor, you should be able to find substitutes for most of the parts. And if you're in a shopping mood, you might be able to find cheaper alternatives. We've tried to select the cheapest suitable option in each case, but prices of course vary over time and at different vendors.

Here are some guidelines for selecting substitutions:

- In all cases, make sure that the physical package is compatible. Make sure the pin or lead wire layout matches, and check the size to make sure the replacement isn't too big to fit the space on the board. Physically smaller parts are usually okay; bigger parts might not fit.
- Resistors with the same resistance (Ohms) value are usually interchangeable. However, if there's also a wattage rating listed, you must use a part with the specified wattage rating or higher. If no wattage is listed in the table, you can assume a default of 1/4W. (Most through-hole resistors are 1/4W or higher anyway, which is why we don't bother mentioning the wattage in those cases.)
- Capacitors with the same capacitance (μF or nF) value **and** the same type ("electrolytic" or "ceramic") are usually interchangeable. "Ceramic" and "disc" capacitors are the same type. Tantalum capacitors are **not** interchangeable with ceramic/disk, even if they have the same capacitance value.
- The NPN and PNP transistors we use can be replaced with most other "small signal switching" transistors. The polarity (NPN or PNP) must always match. Pay attention to the ordering of the "legs", since that can be different even if the physical package looks identical.
- IC chips usually need to be the exact parts listed. In some cases, though, several manufacturers make compatible equivalents. These will generally have the same number with a different letter prefix. For example, there are equivalent xx847 optocoupler chips from several manufacturers, with names like PC847, K847, LTV-847.

More detailed advice on selecting substitute parts can be found in the chapters on the individual component types in our electronics overview section, A Crash Course in Electronics. Start at Chapter 69, Field Guide to Components and follow the links to the chapters on the various components.

Ribbon cables

Some of the connections to and between the expansion boards are most easily handled with "ribbon cables". These are the type of flat, multi-conductor cables that you see inside PCs to connect some of the internal components together.

The places I recommend ribbon cables are:

- The connection between the main board and power board (main board "PWM OUT" to power board "PWM IN")
- The connection between the main board and the chime board ("Chime Out" to "Chime In")
- The connection between the main board and the plunger sensor
- The connection from the main board to your flasher LEDs

Ribbon cabling isn't an absolute requirement for any of these, but I'd recommended it over other options (especially crimp housings), because it's cheaper and easier, and it makes cleaner data connections.

You can buy pre-assembled ribbon cables, but it's hard to find them for anything other than the most common sizes used in PCs. Fortunately, it's fairly easy to build your own. That's cheaper than buying pre-made cables, and it lets you build the exact length you need and with the number of conductors you need.

The parts list assumes that you're going to be building your own custom ribbon cables where needed. As such, it lists the **connectors** needed at the end of each required ribbon cable. However, it doesn't list the **wire**. The wire is a generic part that you can find anywhere, and you'll want to figure out what length you need, so we leave it up to you to select the wire.

See Chapter 83, Ribbon Cables for details on buying the wire and attaching the connectors to the cables.

KL25Z Microcontroller (Standalone)

ID	Description	Mouser #	Notes
FRDM1	KL25Z microcontroller board	841-FREEDOM-KL25Z	
USB1	USB Cable, USB A to Mini-B, 1.8M, black	538-88732-8802	Qty 2; Note 1
J1	2x08 pin header, vertical (0.1")	855-M20-9760846	
J1	2x08 crimp pin wire housing, mates with 855-M20-9760846	855-M20-1070800	Connector
J1	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 16; Connector Pins
J2	2x10 pin header, vertical (0.1")	855-M20-9761046	Connector
J2	2x10 crimp pin wire housing, mates with 855-M20-9761046	855-M20-1071000	Connector
J2	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 20; Connector Pins
J9	2x08 pin header, vertical (0.1")	855-M20-9760846	
J9	2x08 crimp pin wire housing, mates with 855-M20-9760846	855-M20-1070800	Connector
J9	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 16; Connector Pins
J10	2x06 pin header, vertical (0.1")	855-M20-9760646	

J10	2x06 crimp pin wire housing, mates with 855-M20-9760646	855-M20-1070600	Connector
J10	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 12; Connector Pins

Main Board (KL25Z Interface)

ID	Description	Mouser #	Notes
C1	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C2	150nF (0.15uF) capacitor (ceramic, 2.5mm lead spacing)	581-SR205E154MAR	
C4	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C5	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C6	4.7uF capacitor (ceramic, 2.5mm lead spacing)	810-FG14X5R1H475KRT0	
C7	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C8	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C9	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C11	100uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	667-ECA-1AM101I	
C12	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
D1	1N4007 diode	621-1N4007	
FRDM1	2x10 pin socket, vertical (0.1")	855-M20-7831046	
FRDM1	2x08 pin socket, vertical (0.1")	855-M20-7830846	
FRDM1	2x08 pin socket, vertical (0.1")	855-M20-7830846	
FRDM1	2x06 pin socket, vertical (0.1")	855-M20-7830646	
FRDM1	KL25Z microcontroller board	841-FREEDOM-KL25Z	
FRDM1	2x10 pin header, vertical (0.1")	855-M20-9761046	
FRDM1	2x08 pin header, vertical (0.1")	855-M20-9760846	Qty 2
FRDM1	2x06 pin header, vertical (0.1")	855-M20-9760646	
IC1	TLC5940NT PWM controller IC (28-pin DIP)	595-TLC5940NT	Note 2
IC1	28-pin DIP IC socket	571-1-2199298-9	IC Socket; Optional
IC2	TLC5940NT PWM controller IC (28-pin DIP)	595-TLC5940NT	Note 2
IC2	28-pin DIP IC socket	571-1-2199298-9	IC Socket; Optional
IC5	ULN2064BN Quad Darlington array (16-pin DIP)	511-ULN2064B	
IC5	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
IC6	ULN2064BN Quad Darlington array (16-pin DIP)	511-ULN2064B	

IC6	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
IC7	ULN2064BN Quad Darlington array (16-pin DIP)	511-ULN2064B	
IC7	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
IC8	ULN2064BN Quad Darlington array (16-pin DIP)	511-ULN2064B	
IC8	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
IC11	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC12	LD1117AV33 voltage regulator, 3.3V	511-LD1117AV33	
JP1	2x13 pin header, vertical (0.1")	855-M20-9761346	
JP1	2x13 pin wire housing (0.1"), mates with 855-M20-9761346	649-65239-013LF	Connector
JP1	Crimp pin for Amphenol FCI 65239-xxxx wire housing	649-76357-301LF	Qty 26; Connector Pins
JP2	2x04 pin header, vertical (0.1")	855-M20-9760446	
JP3	2x02 pin header, vertical (0.1")	855-M20-9760246	
JP4	2x03 pin header, vertical (0.1")	855-M20-9980346	
JP4	2x03 crimp pin wire housing, mates with 855-M20-9760346	855-M20-1070300	Connector
JP4	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 6; Connector Pins
JP5	2x05 pin header, vertical (0.1")	855-M20-9760546	
JP6	2x05 pin header, vertical (0.1")	855-M20-9760546	
JP7	2-pin header (0.1") with friction lock	538-22-23-2021	
JP7	2-pin wire housing (0.1"), mates with 538-22-23-2021; 2695 series	538-22-01-2021	Connector
JP7	2.54mm crimp terminals for Molex 2695 series wire housings	538-08-50-0114	Qty 2; Connector Pins
JP8	2x09 pin header, vertical (0.1")	649-77313-101-18LF	
JP8	2x09 crimp pin wire housing, mates with 855-M20-9760946	855-M20-1070900	Connector
JP8	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 18; Connector Pins
JP9	2-pin header (0.1") with friction lock	538-22-23-2021	Note 3
JP9	2-pin wire housing (0.1"), mates with 538-22-23-2021; 2695 series	538-22-01-2021	Connector
JP9	2.54mm crimp terminals for Molex 2695 series wire housings	538-08-50-0114	Qty 2; Connector Pins
JP9	1x02 pin header, vertical (0.1")	855-M20-9990246	Note 3; Alternate
JP9	1x02 crimp pin wire housing, mates with 855-M20-9990246	855-M20-1060200	Alternate; Connector
JP9	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 2; Alternate; Connector Pins
JP10	4-pin header (3.96mm) with friction lock	538-26-60-4040	
JP10	4-pin wire housing (3.96mm), mates with 538-26-60-4040; 2139 series	538-09-50-3041	Connector
JP10	3.96mm crimp terminals for Molex 2139 series wire housings	538-08-50-0106	Qty 4; Connector Pins

JP11	2x08 pin header, vertical (0.1")	855-M20-9760846	
JP11	2x08 pin IDC socket, mates with 0.1" pin header	164-9008-E	Qty 2; Connector
JP12	2x02 pin header, vertical (0.1")	855-M20-9760246	
JP12	2x02 crimp pin wire housing, mates with 855-M20-9760246	855-M20-1070200	Connector
K1	Omron G5V-2 PCB relay	653-G5V-2-DC5	
OK1	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK1	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK2	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK2	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK3	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK3	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK4	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK4	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK5	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK13	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK14	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK15	PC817 optocoupler (4-pin DIP)	859-LTV-817	
Q1	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
R1	10K resistor	660-MS1/4DCT52R1002	
R2	4K resistor	660-MF1/4DCT52R4021F	
R3	2.2K resistor	660-MFS1/4DCT52R2201	
R4	10K resistor	660-MS1/4DCT52R1002	
R5	2.2K resistor	660-MFS1/4DCT52R2201	Note 5
R5	4K resistor	660-MF1/4DCT52R4021F	Note 5; Alternate
R5	1.3K resistor	660-MF1/4DCT52R1301F	Note 5; Alternate
R5	1K resistor	660-MS1/4DCT52R1001	Note 5; Alternate
R5	787 ohm resistor	660-MF1/4DCT52R7870F	Note 5; Alternate
R5	680 ohm resistor	660-MF1/4DCT52R6800F	Note 5; Alternate
R6	1M ohm resistor (7mm lead spacing)	660-MFS1/4DCT52R1004	
R7	220 ohm resistor	660-MF1/4DCT52R2200F	
R8	2.2M ohm resistor (7mm lead spacing)	270-2.2M-RC	
R9	2.2K resistor	660-MFS1/4DCT52R2201	

R10	100K ohm resistor (7mm lead spacing)	660-MFS1/4DC1003F	
R11	27 ohm resistor, 1/2W	279-LR1F27R	Note 6
R11	39 ohm resistor, 1/2W	594-5073NW39R00J	Note 6; Alternate
R12	100K ohm resistor (7mm lead spacing)	660-MFS1/4DC1003F	
R13	47 ohm resistor (7mm lead spacing)	660-MFS1/4DCT52R47R0	
R14	1K resistor	660-MS1/4DCT52R1001	
R16	10K resistor	660-MS1/4DCT52R1002	
R18	47 ohm resistor (7mm lead spacing)	660-MFS1/4DCT52R47R0	
R36	560 ohm resistor	594-SFR16S0005600FR5	
R37	2.2K resistor	660-MFS1/4DCT52R2201	
R38	220 ohm resistor	660-MF1/4DCT52R2200F	
R39	560 ohm resistor	594-SFR16S0005600FR5	
R40	10K resistor	660-MS1/4DCT52R1002	
R41	4.7K resistor	660-MS1/4DCT52R4701	
R42	2.2K resistor	660-MFS1/4DCT52R2201	
R43	82 ohm resistor	603-MFR-25FRF52-82R	
R44	4.7K resistor	660-MS1/4DCT52R4701	
R46	10K resistor	660-MS1/4DCT52R1002	
R47	22K resistor	660-MF1/4LCT52R223G	
R49	2.2K resistor	660-MFS1/4DCT52R2201	
T1	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T2	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T3	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T4	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T5	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T6	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T7	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T8	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7

U\$2	TSOP384 Infrared receiver, 38kHz	78-TSOP38438	
USB1	USB Cable, USB A to Mini-B, 1.8M, black	538-88732-8802	Qty 2; Note 1
LED1	Infrared (IR) emitter LED, 100mA, 25 degree beam	782-TSAL6400	Qty 2; Note 8

Power Board

ID	Description	Mouser #	Notes
C1	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C2	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C6	4.7uF capacitor (ceramic, 2.5mm lead spacing)	810-FG14X5R1H475KRT0	
C11	100uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	667-ECA-1AM101I	
C12	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
IC1	TLC5940NT PWM controller IC (28-pin DIP)	595-TLC5940NT	Note 2
IC1	28-pin DIP IC socket	571-1-2199298-9	IC Socket; Optional
IC2	TLC5940NT PWM controller IC (28-pin DIP)	595-TLC5940NT	Note 2
IC2	28-pin DIP IC socket	571-1-2199298-9	IC Socket; Optional
IC12	LD1117AV33 voltage regulator, 3.3V	511-LD1117AV33	
JP1	2-pin header (0.1") with friction lock	538-22-23-2021	
JP1	2-pin wire housing (0.1"), mates with 538-22-23-2021; 2695 series	538-22-01-2021	Connector
JP1	2.54mm crimp terminals for Molex 2695 series wire housings	538-08-50-0114	Qty 2; Connector Pins
JP2	2x05 pin header, vertical (0.1")	855-M20-9760546	
JP2	2x05 pin IDC socket, mates with 0.1" pin header	164-9006-E	Qty 2; Connector
JP3	2x05 pin header, vertical (0.1")	855-M20-9760546	
JP4	4-pin header (3.96mm) with friction lock	538-26-60-4040	
JP4	4-pin wire housing (3.96mm), mates with 538-26-60-4040; 2139 series	538-09-50-3041	Connector
JP4	3.96mm crimp terminals for Molex 2139 series wire housings	538-08-50-0106	Qty 4; Connector Pins
JP5	1x16 pin header, vertical (0.1")	855-M20-9991646	
JP5	1x08 crimp pin wire housing, mates with 855-M20-9990846	855-M20-1060800	Qty 2; Connector
JP5	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 16; Connector Pins
JP6	1x16 pin header, vertical (0.1")	855-M20-9991646	
JP6	1x08 crimp pin wire housing, mates with 855-M20-9990846	855-M20-1060800	Qty 2; Connector
JP6	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 16; Connector Pins
OK1	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	

OK1	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK2	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK2	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK3	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK3	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK4	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK4	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK5	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK5	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK6	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK6	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK7	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK7	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
OK8	PC847 quad optocoupler (16-pin DIP)	859-LTV-847	
OK8	16-pin DIP IC socket	571-1-2199298-4	IC Socket; Optional
Q1A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q1B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q1C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q1D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q2A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q2B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q2C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q2D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q3A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q3B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q3C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q3D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q4A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q4B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4

Q4C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q4D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q5A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q5B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q5C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q5D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q6A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q6B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q6C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q6D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q7A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q7B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q7C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q7D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q8A	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q8B	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q8C	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q8D	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
R1	10K resistor	660-MF1/4LCT52R103J	
R2	4K resistor	660-MF1/4DCT52R4021F	
R3	4K resistor	660-MF1/4DCT52R4021F	
R4	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0	
R5	1K resistor	660-MS1/4DCT52R1001	
R6	1K resistor	660-MS1/4DCT52R1001	
R7	1K resistor	660-MS1/4DCT52R1001	
R8	1K resistor	660-MS1/4DCT52R1001	

R9	1K resistor	660- MS1/4DCT52R1001
R10	1K resistor	660- MS1/4DCT52R1001
R11	1K resistor	660- MS1/4DCT52R1001
R12	1K resistor	660- MS1/4DCT52R1001
R13	1K resistor	660- MS1/4DCT52R1001
R14	1K resistor	660- MS1/4DCT52R1001
R15	1K resistor	660- MS1/4DCT52R1001
R16	1K resistor	660- MS1/4DCT52R1001
R17	1K resistor	660- MS1/4DCT52R1001
R18	1K resistor	660- MS1/4DCT52R1001
R19	1K resistor	660- MS1/4DCT52R1001
R20	1K resistor	660- MS1/4DCT52R1001
R21	1K resistor	660- MS1/4DCT52R1001
R22	1K resistor	660- MS1/4DCT52R1001
R23	1K resistor	660- MS1/4DCT52R1001
R24	1K resistor	660- MS1/4DCT52R1001
R25	1K resistor	660- MS1/4DCT52R1001
R26	1K resistor	660- MS1/4DCT52R1001
R27	1K resistor	660- MS1/4DCT52R1001
R28	1K resistor	660- MS1/4DCT52R1001
R29	1K resistor	660- MS1/4DCT52R1001
R30	1K resistor	660- MS1/4DCT52R1001
R31	1K resistor	660- MS1/4DCT52R1001
R32	1K resistor	660- MS1/4DCT52R1001
R33	1K resistor	660- MS1/4DCT52R1001
R34	1K resistor	660- MS1/4DCT52R1001

R35	1K resistor	660- MS1/4DCT52R1001
R36	1K resistor	660- MS1/4DCT52R1001
R37	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R38	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R39	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R40	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R41	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R42	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R43	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R44	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R45	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R46	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R47	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R48	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R49	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R50	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R51	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R52	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R53	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R54	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R55	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R56	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R57	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R58	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R59	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0
R60	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0

R61	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R62	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R63	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R64	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R65	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R66	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R67	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0

Chime Board

ID	Description	Mouser #	Notes
C1	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C2	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C3	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C4	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C5	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C6	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C7	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C8	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C9	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C10	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C11	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C12	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C13	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C14	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C15	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C16	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C17	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	

C18	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C19	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C20	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C21	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C22	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C23	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C24	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C25	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C26	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C27	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C28	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C29	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C30	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C31	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
C32	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
C33	1uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	80-ESK105M100AC3FA	
IC1	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC2	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC3	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC4	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC5	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC6	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC7	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
IC8	74HC595 8-bit shift register (16-pin DIP)	595-SN74HC595N	
IC9	ICM7555 timer IC (8-pin DIP)	968-ICM7555IPAZ	
JP1	2x05 pin header, vertical (0.1")	855-M20-9760546	
JP1	2x05 pin IDC socket, mates with 0.1" pin header	164-9006-E	Qty 2; Connector
JP2	2x05 pin header, vertical (0.1")	855-M20-9760546	
JP7	2-pin header (0.1") with friction lock	538-22-23-2021	
JP7	2-pin wire housing (0.1"), mates with 538-22-23-2021; 2695 series	538-22-01-2021	Connector
JP7	2.54mm crimp terminals for Molex 2695 series wire housings	538-08-50-0114	Qty 2; Connector Pins
JP9	1x08 pin header, vertical (0.1")	855-M20-9990846	

JP9	1x08 crimp pin wire housing, mates with 855-M20-9990846	855-M20-1060800	Connector
JP9	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 8; Connector Pins
JP10	4-pin header (3.96mm) with friction lock	538-26-60-4040	
JP10	4-pin wire housing (3.96mm), mates with 538-26-60-4040; 2139 series	538-09-50-3041	Connector
JP10	3.96mm crimp terminals for Molex 2139 series wire housings	538-08-50-0106	Qty 4; Connector Pins
OK1	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK2	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK3	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK4	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK5	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK6	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK7	PC817 optocoupler (4-pin DIP)	859-LTV-817	
OK8	PC817 optocoupler (4-pin DIP)	859-LTV-817	
Q1	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q2	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q3	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q4	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q5	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q6	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q7	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
Q8	FQP13N06L N-channel MOSFET (TO-220 package)	512-FQP13N06L	Note 4
R1	1M resistor	660-MFS1/4DCT52R1004	
R2	2.2M resistor	660-MF1/4LCT52R225G	
R3	100K resistor	660-MFS1/4DC1003F	
R4	100K resistor	660-MFS1/4DC1003F	
R5	47 ohm resistor	660-MFS1/4DCT52R47R0	
R6	1M resistor	660-MFS1/4DCT52R1004	
R7	1K resistor	660-MS1/4DCT52R1001	
R8	2.2M resistor	660-MF1/4LCT52R225G	
R9	2.2K resistor	660-MFS1/4DCT52R2201	
R10	100K resistor	660-MFS1/4DC1003F	
R11	100K resistor	660-MFS1/4DC1003F	

R12	47 ohm resistor	660-MFS1/4DCT52R47R0
R13	1K resistor	660-MS1/4DCT52R1001
R14	2.2K resistor	660-MFS1/4DCT52R2201
R15	1M resistor	660-MFS1/4DCT52R1004
R16	2.2M resistor	660-MF1/4LCT52R225G
R17	100K resistor	660-MFS1/4DC1003F
R18	100K resistor	660-MFS1/4DC1003F
R19	47 ohm resistor	660-MFS1/4DCT52R47R0
R20	1K resistor	660-MS1/4DCT52R1001
R21	2.2K resistor	660-MFS1/4DCT52R2201
R22	1M resistor	660-MFS1/4DCT52R1004
R23	2.2M resistor	660-MF1/4LCT52R225G
R24	100K resistor	660-MFS1/4DC1003F
R25	100K resistor	660-MFS1/4DC1003F
R26	47 ohm resistor	660-MFS1/4DCT52R47R0
R27	1K resistor	660-MS1/4DCT52R1001
R28	2.2K resistor	660-MFS1/4DCT52R2201
R29	1M resistor	660-MFS1/4DCT52R1004
R30	2.2M resistor	660-MF1/4LCT52R225G
R31	100K resistor	660-MFS1/4DC1003F
R32	100K resistor	660-MFS1/4DC1003F
R33	47 ohm resistor	660-MFS1/4DCT52R47R0
R34	1K resistor	660-MS1/4DCT52R1001
R35	2.2K resistor	660-MFS1/4DCT52R2201
R36	47 ohm resistor (5mm lead spacing)	660-MFS1/4DCT52R47R0
R37	1M resistor	660-MFS1/4DCT52R1004
R38	47 ohm resistor	660-MFS1/4DCT52R47R0
R39	1K resistor	660-MS1/4DCT52R1001
R40	47 ohm resistor	660-MFS1/4DCT52R47R0

R41	1K resistor	660- MS1/4DCT52R1001	
R42	47 ohm resistor	660- MFS1/4DCT52R47R0	
R43	1K resistor	660- MS1/4DCT52R1001	
R44	1M resistor	660- MFS1/4DCT52R1004	
R45	2.2M resistor	660- MF1/4LCT52R225G	
R46	100K resistor	660-MFS1/4DC1003F	
R47	100K resistor	660-MFS1/4DC1003F	
R48	2.2K resistor	660- MFS1/4DCT52R2201	
R49	2.2M resistor	660- MF1/4LCT52R225G	
R50	100K resistor	660-MFS1/4DC1003F	
R51	100K resistor	660-MFS1/4DC1003F	
R52	2.2K resistor	660- MFS1/4DCT52R2201	
R53	1M resistor	660- MFS1/4DCT52R1004	
R54	2.2M resistor	660- MF1/4LCT52R225G	
R55	100K resistor	660-MFS1/4DC1003F	
R56	100K resistor	660-MFS1/4DC1003F	
R57	2.2K resistor	660- MFS1/4DCT52R2201	
R58	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
R59	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
R60	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
R61	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
R62	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
R63	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
R64	47 ohm resistor (5mm lead spacing)	660- MFS1/4DCT52R47R0	
T1	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T2	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T3	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T4	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T5	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T6	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T7	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T8	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T9	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7

T10	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T11	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T12	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T13	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T14	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7
T15	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
T16	2N4403 PNP transistor (TO92-EBC package)	512-2N4403TFR	Note 7

Plunger Sensor (TSL1410R linear photo sensor array)

Note: The TSL1410/12 sensors were discontinued by the manufacturer in 2016 and are no longer available anywhere that I'm aware of. This section is therefore of historical interest only. I'm keeping it in the guide for the sake of completeness, and on the off chance that someone discovers a dusty old carton full of unsold TSL1410's buried at the back of a shelf in a warehouse somewhere, and puts them up for sale on eBay, or the even more remote chance that the manufacturer does another run at some point. But for now, I'm sorry to say that these devices can't be bought at any price, so new cabinet builders will have to look to the other plunger sensor options instead.

ID	Description	Mouser #	Notes
U1	AMS TSL1410R linear photo sensor array, 1280-pixel	856-TSL1410R	
<i>U1</i>	<i>AMS TSL1412S linear photo sensor array, 1536-pixel</i>	<i>856-TSL1412S</i>	<i>Note 9; Alternate</i>
C1	100nF (0.1uF) capacitor (ceramic, 2.5mm lead spacing)	21RZ310-RC	
JP1	Ribbon cable PCB connector, 2x4 position	538-91577-1308	Note 10; Note 11
JP2	2x04 pin IDC socket, mates with 0.1" pin header	710-61200823021	Note 10

Plunger Sensor (Potentiometer)

ID	Description	Mouser #	Notes
R1	Slide potentiometer, 10K ohm, 100mm travel	688-RSA0N11S9A0K	
JP2	2x04 crimp pin wire housing, mates with 855-M20-9760446	855-M20-1070400	Exp. Boards Only; Connector
JP2	0.1" crimp pins for M20-106xxxx/107xxxx wire housings	855-M20-1160042	Qty 8; Exp. Boards Only; Connector Pins

Plunger Sensor (AEDR-8300 optical encoder)

ID	Description	Mouser #	Notes
JP1	Ribbon cable PCB connector, 2x4 position	538-91577-1308	Note 10
JP2	2x04 pin IDC socket, mates with 0.1" pin header	710-61200823021	Note 10
OK1	AEDR-8300 optical encoder, 75 LPI	630-AEDR-8300-1K2	
R1	220 ohm resistor	660-MF1/4DCT52R2200F	
R2	2.7K resistor	660-MFS1/4DCT52R2701	
R3	2.7K resistor	660-MFS1/4DCT52R2701	

Plunger Sensor (TCD1103 linear image sensor)

ID	Description	Mouser #	Notes
C1	0.1uF capacitor, 25V, MLCC, SMD 0402	77-VJ0402Y104KXXCWBC	
C2	10uF capacitor, 25V, aluminum electrolytic, SMD	710-865080440002	
IC1	Toshiba TCD1103GFG linear image sensor	757-TCD1103GFG8ZAA	
IC2	74HC04D hex inverter	771-HC04D652	
JP1	Ribbon cable PCB connector, 2x4 position	538-91577-1308	Note 10
JP2	2x04 pin IDC socket, mates with 0.1" pin header	710-61200823021	Note 10
Q1	2SA1162-Y PNP transistor	757-2SA1162-YLF	
R1	150 Ohm resistor, SMD 0805	603-RC0805FR-07150RL	
R2	150 Ohm resistor, SMD 0805	603-RC0805FR-07150RL	
R3	2.2K resistor, SMD 8085	603-RC0805FR-072K2L	

Plunger Calibration Button (Standalone)

ID	Description	Mouser #	Notes
S1	Illuminated momentary pushbutton switch, blue LED	612-WBL2UOABQR05CLR	
Q1	2N4401 NPN transistor (TO92-EBC package)	512-2N4401TFR	Note 7
R1	2.2K resistor	660-MFS1/4DCT52R2201	
R2	82 ohm resistor	603-MFR-25FRF52-82R	

Plunger Calibration Button (Expansion Boards)

ID	Description	Mouser #	Notes
S1	Illuminated momentary pushbutton switch, blue LED	612-WBL2UOABQR05CLR	
JP3	2x02 crimp pin wire housing, mates with 855-M20-9760246	855-M20-1070200	Connector

Williams Coin Door 13-pin Connector Board - version 2

ID	Description	Mouser #	Notes
C1	10uF electrolytic capacitor	667-ECA-1EM100I	
C2	10uF electrolytic capacitor	667-ECA-1EM100I	
IC1	LD1086BV-DG adjustable voltage regulator	511-LD1086BV-DG	
JP1	Phoenix Contact screw terminal, 9 positions, 0.1" pitch	651-1725724	
JP2	Phoenix Contact screw terminal, 4 positions, 0.1" pitch	651-1725672	
JP3	Molex 13-pin header, 3.96mm pitch	538-26-60-4130	
R1	120 Ohm resistor	660-MFS1/4DCT52R1200	
R2	487 ohm resistor	603-MFR-25FBF52-487R	

Williams Coin Door 13-pin Connector Board - version 1

ID	Description	Mouser #	Notes
C1	0.33uF electrolytic capacitor	710-860010772003	
C11	22uF capacitor (electrolytic, vertical, 2.5mm lead spacing)	667-ECA-1EM220I	
JP1	1x10 pin header, vertical (0.1")	855-M20-9991046	

JP2	1x02 pin header, vertical (0.1")	855-M20-9990246
JP3	1x02 pin header, vertical (0.1")	855-M20-9990246
Q1	6.3V voltage regulator, 1.5A	513-NJM2396F63
X1	Molex 13-pin header, 3.96mm pitch	538-26-60-4130

Note 1: You only need one USB cable to connect the KL25Z during routine use, but I recommend getting two. This lets you leave the programming port plugged in all the time, so you can install firmware updates without having to open up your cabinet and move cables around.

Note 2: Be aware that **you can't buy the TLC5940NT at Mouser** or other mainstream electronics suppliers, but you can still buy it on eBay and Aliexpress. Texas Instruments stopped manufacturing the chip many years ago, and the supply of genuine parts has long since dried up. Fortunately, that doesn't stop the "gray market" sellers on eBay and Aliexpress, and the chip continues to be readily available in those venues. The chip has been out of official production for so long that the ones you can buy now are almost certainly unauthorized knockoffs, and (as you might expect) the Dead-On-Arrival rate is reportedly pretty high lately. You might want to order extras in case you get bad chips in your batch. Unfortunately, there's no similar through-hole chip currently available to use as a substitute, so we're stuck for now with the sometimes problematic knockoffs.

Note 3: This part is listed with a Molex connector, with a plain 1x2 pin header also included as an alternative. You can use either one, but the Molex connector is recommended because it has a friction lock that keeps the connected cable in place more securely. Plain 2-pin connectors in this size tend to come loose easily in a high-vibration environment like a pin cab.

Note 4: Any similar N-channel MOSFET can be substituted, so you can choose something based on price and availability. This part is used for logic circuit power switching, which is a basic task that many MOSFETs can perform. Substitutions must have the same physical package (TO-220-3) and pin layout (pins in G-D-S order) so that they'll fit the space on the PCB. In terms of electrical specs, the key features are low on-state resistance, in the 100 milli-Ohm range or below, and drain voltage (V_{DS}) and current (I_D) limits high enough for the output devices you plan to attach. We drive the gate with 12V, so the device must be fully switched on at 12V and $V_{GS_{max}}$ must be at least 12V. We recommend drain current limits of at least 6A @ 30V. These specs are fairly undemanding, so you should be able to find many options. Recommended alternatives: FQP30N06L (or almost anything in the FQPxxN06L series), BUK7575-55A.

Note 5: This resistor sets the current limit for the RGB flipper button LED outputs. Select a resistor according to the forward current of your LEDs. If you're using Lightmite boards with two RGB LEDs, set this for 60mA.

- 10mA → 4K ohm resistor
- 20mA → 2.2K
- 30mA → 1.3K
- 40mA → 1K
- 50mA → 780 ohms
- 60mA → 680 ohms (use for Lightmites)

Note 6: Select a resistor according to the IR LED(s) used for the TV ON feature, if any. For the reference 100mA LED:

- For one LED: use 39 ohms 1/2 Watt
- For two LEDs in series: use 27 ohm 1/2 Watt

Omit the resistor if no LED will be connected.

Note 7: If you're in Europe, it might be more convenient to substitute transistor type BC337 for each 2N4401, and type BC327 for each 2N4403. The BC3xx types are reportedly easier to find in Europe. See Chapter 92, European Transistor Substitutions.

Note 8: Note the quantity: you can connect 1 or 2 of these IR LEDs according to your needs. One is usually sufficient, but you might need two if you're controlling multiple TVs. This gives you more flexibility positioning the transmitters so that they can reach the remote receivers in all of the TVs.

Note 9: The 1410R and 1412S are almost identical sensors. The Pinscape software is compatible with both. The key difference is that the 1412S has more pixels, which makes it slightly longer, as the pixel pitch is the same in both devices. The greater length of the 1412S allows more margin of error positioning the sensor, but it also means it takes up more space in the cabinet, which might constrain positioning for the playfield TV. Also, the 1412S is usually more expensive than the 1410R.

Note 10: This part is only needed if you plan to use a ribbon cable to connect the sensor to the expansion boards, or to the Plunger Breakout Board if you're using a standalone KL25Z. The ribbon cable is optional, as you can use hook-up wire instead. But I recommend the ribbon cable; it's easier to assemble and provides a cleaner data signal connection.

Note 11: This part is only needed if you're going to use my adapter circuit board (described in the plunger chapter), and you're going to use a ribbon cable to connect the sensor to the expansion boards or the Plunger Breakout Board. You can use individual hookup wires instead if you prefer, although the ribbon cable is easier if you're using the adapter board, and it provides a cleaner signal path for the high-speed data signals used by this sensor.

Alternate: This part is an optional substitution for the other listed part(s) for the same item. See the item notes for details.

Exp. Boards Only: This part is only needed if you're using the expansion boards. It's not needed for a stand-alone KL25Z setup.

Connector: A plug or socket designed to connect to a mating pin header (or the like) on one of the circuit boards. These are usually listed in the sections for the **connected** devices rather than with the circuit boards that they plug into. For example, the connector that plugs into the "plunger" header on the expansion board is listed in the parts section for the plunger rather than with the expansion board. We group them this way so that you only have to buy parts for the sections you're actually building.

Connector Pins: This is a set of contact terminals needed as part of the connector listed for the item.

IC Socket: Optional socket for IC. You can solder the IC directly to the circuit board if you prefer, or you can use a socket. The main benefit of a socket is that it makes it easy to replace a defective or damaged chip. Chips soldered directly to the board can be quite difficult to remove.

92. European Transistor Substitutions

If you're in Europe, you might find it convenient and cost-saving to use different transistors from the ones specified in the parts list.

The expansion boards use two types of transistors, which are known by their part numbers, 2N4401 and 2N4403. We chose these because they're inexpensive and easy to find in the US. There's nothing else particularly special about them; we use them for simple logic switching, which is a job that many other similar transistors can perform equally well.

If you're in Europe, the 2N440x series is reportedly hard to find, and importing them from the US is expensive in terms of postage and customs fees. Fortunately, these parts have equivalents that are readily available in Europe:

- 2N4401 → BC337
- 2N4403 → BC327

You can simply make these substitutions everywhere you see the 2N440x parts on the circuit boards. Be sure to replace each 2N440x with the **corresponding** BC3xx device as shown above, since these are complementary pairs (NPN and PNP).

Attention! Reversed Leads!

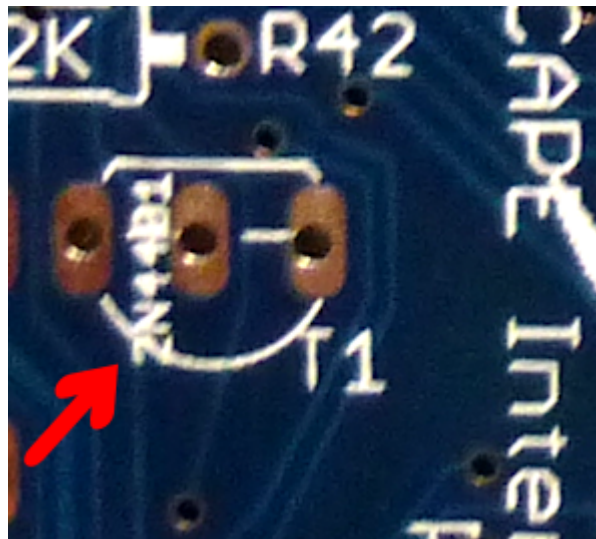
If you do make these substitutions, there's a **really important detail**: the order of the leads is reversed on the BC3xx versus the 2N440x. This isn't a big problem, though. If you use the BC3xx parts, simply rotate the device 180 degrees from the 2N440x part orientation. That'll put the leads in the right order for the board wiring.

But there's yet another complication! It's a complication that's designed to make your life easier, but it's still something to be aware of.

There are **EU** versions of the expansion boards that are marked properly for the BC3xx parts. These are **electrically** identical to the US boards, but the **silkscreen printing** on the board is drawn for the BC3xx parts. If you have the EU boards and you're using the BC3xx parts, the orientation marker on the board is already correct, so you don't have to rotate the parts from what's shown on the board.

If you're not sure if you have US or EU boards, check the tiny text printed inside the transistor outlines on the board. This will tell you the type of transistor that the board is marked for - it will show either "2N440x" or "BC3xx". The text really is tiny, so you might need a magnifying glass.

If you see 2N440x printed there, you have a US board, so you'll need to **rotate your BC3xx parts 180°** from the outline printed on the board.



If you see BC3xx printed on the board, you have an EU board, so simply install the BC3xx exactly as shown on the board.

93. Fabricating the Expansion Boards

The Pinscape expansion boards were designed using a software package called EAGLE, from Autodesk. One of the things EAGLE can do is to convert a circuit plan into a set of Computer Aided Manufacturing files that you can upload to a PCB manufacturer to have a physical board made from the plans.

This section explains how to do that, so that you can have your own copies of the Pinscape boards made by a PCB maker.

You can also use this process to create custom versions of the Pinscape boards with any modifications you want to make. The Pinscape EAGLE files are all open-source, meaning you're free to customize them as needed, and free to manufacture your own versions. EAGLE itself is available in a free hobbyist version; see the Autodesk site for downloads. (The free version of EAGLE has some limitations; if you need a less restricted version, Autodesk sells EAGLE on a monthly subscription basis, so you can use it for a small job without a huge investment.)

Where to download the board plans

The EAGLE plans for the Pinscape boards are available here:

mjrnet.org/pinscape/expansion_board/download.php

That page may show multiple versions. I'd recommend downloading the latest "Release" version listed, unless you have a reason to use one of the older releases. The "latest working snapshot" version reflects work in progress, so it might have changes that haven't been finalized or tested yet. It's usually better to use a version specifically identified as a release version.

EAGLE files in the downloads

In the board plan downloads, you'll find two main EAGLE file types:

- **.sch** files are the schematics for the boards. A schematic is an abstract circuit plan showing the components and their connections, without any information on how they're physically arranged on a board.
- **.brd** files represent the physical layouts for the printed circuit boards, showing how the parts are arranged, where all the holes are drilled, and how the copper traces on the board are laid out.

The .sch and .brd files are always paired. When you open one of the other type of file in EAGLE, it will look for the counterpart and open it as well, to keep the two in sync.

Other files in the downloads

- **BOM.csv** files are spreadsheets (in comma-separated value or CSV format) with the "Bill of Materials" for the boards. These were generated by EAGLE from the schematics to list all of the parts used in the schematics. These can be useful for uploading to a vendor like Mouser or DigiKey to order the parts needed to populate the board. (These can also be used with PCB manufacturers who can assemble the complete boards, but don't get your hopes up that you'll be able to get a PCB maker to assemble the boards for, as this service is prohibitively expensive unless you're ordering large quantities of the boards.) You can open these files in a spreadsheet program like Excel. This is only for reference; to place an order, it's better to use the Chapter 91, Electronic Parts List in this guide.

- **MouserCart.csv** is another CSV file that contains a version of the parts list in the Mouser shopping cart export format. This is only for reference; to place an order, it's better to use the Chapter 91, Electronic Parts List in this guide.
- **Schematic.pdf** contains a print-out of the schematics in PDF format, for viewing in Adobe Acrobat or a Web browser. These are provided in case you want to look at the schematics without bothering to install EAGLE.
- **Layout snapshot.jpg** contains a JPG screen image of the board layout from EAGLE, so that you can view these without installing EAGLE.
- **Gerbers (Elecrow).zip** contains the Gerber files (see below) generated for Elecrow.com, which is the manufacturer I've been using to make copies of the boards. See below for an explanation of what the Gerber files are. You can use these files to place your own order from Elecrow without going through the whole Gerber generation process. You probably can't use these with any other vendor, since each vendor has its own design rules that you should use to generate a new set of Gerbers specific to your chosen vendor.

Selecting a vendor

There's a nice online tool for selecting a PCB manufacturer for a project:

pcbshopper.com

If you want to shop around for the best price, go there and plug in the appropriate parameters for the Pinscape boards:

- 10x10 cm (be sure to select centimeters, not inches)
- 2 layers
- Any mask color you like (it makes no functional difference)
- Top silkscreen (two-sided is not required)
- Cheapest surface finish
- Cheapest board thickness, usually 1.6mm
- Cheapest copper weight, usually 1 oz
- Minimum trace width 6 mm
- Minimum drill 16 mil
- No stencil required
- No quality certifications required
- 1 design

For quantity, the most common minimum order size is 10 copies, but some go as low as 5 copies. You can try pricing with 5 and 10 copies to check options. It might be cheaper to order 10 copies than 5, paradoxically, so I'd try both options even if you don't actually need 10 copies.

For any of the options you're not sure about, you can just leave them with the default settings. The Pinscape boards don't have any unusual requirements. The default options offered by most manufacturers will work just fine.

Recommendations

I've used Elecrow for most of my PCB orders. Elecrow is a Chinese electronics e-tailer with a decent Web site and good prices on PCB manufacturing in small quantities for hobbyists. The board quality for my orders has been consistently good. I've heard

from a couple of other people who weren't entirely happy with Elecrow's quality, although their complaints weren't about severe problems, just that the manufacturing precision wasn't up to their expectations. So far I've been satisfied that Elecrow's product is precise enough for the Pinscape boards (which aren't extremely dense by modern standards).

There are numerous other Chinese manufacturers with prices and capabilities similar to Elecrow's. There seems to be a large enough hobbyist market that many vendors offer small-quantity production at low prices. There are also some American and European fabricators. The prices at the Chinese companies are much lower than their Western counterparts, although shipping prices from China are quite high; in most of my Elecrow orders, the shipping charges have been on par with or even higher than the manufacturing cost. You should also take into account any import duties you'll have to pay in your country. (If you're in the United States, you don't have to pay any import duties on goods received below \$200 per day.) If you can find a domestic company, you might save enough on shipping and duty charges to make up for the higher PCB prices.

One American company I'll call out specifically is OSH Park. They make it extremely easy to order, by letting you upload an EAGLE .brd file directly, without going through the Gerber generation process described below. They have great, fast service, free shipping, and a minimum order size of only 3 copies of a board design. They're fantastic for prototyping small boards. The downside is that they're quite expensive for larger boards like the Pinscape expansion boards. I'd highly recommend them for tiny boards, such as the sensor board for the Chapter 104, AEDR-8300 plunger - that only costs about \$5 to make at OSH Park, including shipping.

PCB ordering process

Once you've selected a vendor, you can follow the basic process below to order copies of the Pinscape PCBs. There might be some variations at different vendors, so be sure to read your chosen vendor's instructions as well.

- Select the PCB manufacturing options
 - 10x10 cm (be sure to select centimeters, not inches)
 - 2 layers
 - Any mask color you like (it makes no functional difference)
 - Cheapest surface finish
 - Board thickness 1.6mm, or cheapest
 - Copper weight 1 oz, or any higher value if it's cheaper (but don't go below 1 oz)
 - No stencil required
- Upload the design files that the vendor requires:
 - If you're using Elecrow, you can upload the appropriate **xxx Gerber (Elecrow).zip** files from the Pinscape downloads. Elecrow wants you to upload the .zip file itself, so there's no need to unpack it on your system.
 - If you're using a vendor other than Elecrow, and they require Gerber format files, generate a set of Gerbers as described below and upload those. Most vendors will want you to combine all of the Gerber output files into a single .zip file before uploading.
 - If your vendor accepts EAGLE **.brd** files, simply upload the appropriate .brd file from the Pinscape downloads.

Note! For the Expansion Boards only, you might want to upload Gerber files, generated using the procedure below, *even if* your vendor accepts .brd files directly. The reason is that the Expansion Boards have some custom printing layers for the silkscreen (which controls the white text printed on the board, showing the part names and outlines and so forth), which the PCB vendor probably won't include by default if you just give them the .brd files. Those extra printing layers are purely cosmetic, so the boards will still function the same way without them, but it might be harder to figure out which parts go where without the extra guide markings. This only applies to the expansion boards; I don't think any of my other board designs (such as the plunger interface boards) have any special printing layers, so those should all be fine to upload as .brd files.

- Place the order!

Generating vendor-specific Gerbers

Most PCB manufacturers require you to submit the plans using a file format known as Gerber files. These files describe the various elements that go into manufacturing the boards, such as the placement of the drill holes, copper traces, and silkscreen markings.

The thing that's a little tricky about Gerber files is that they contain manufacturer-specific information. That means that the Gerber files have to be created specially for each manufacturer. That's why the Gerber files included in the Pinscape downloads are specifically labeled as Elecrow Gerbers: these are the Gerbers I use to submit orders to Elecrow, and they're only for Elecrow.

Fortunately, EAGLE provides an automated to generate the Gerber files for a given PCB maker. EAGLE's .sch and .brd files are universal, so we do all of our design work in that format so that it can be used with any PCB maker. When it comes time to actually manufacture the boards, we pick our PCB maker, and then use EAGLE's generator process to convert the .brd file into a Gerber specifically for our chosen manufacturer.

Here's the procedure:

- Download EAGLE. Autodesk offers a free hobbyist version that you can use for this process.
- Choose your PCB maker.
- On your PCB maker's Web site, read their instructions for submitting orders. As part of this, they should provide an EAGLE **.cam** file that you can download. Do so. The **.cam** file is the key to generating the Gerbers. Some vendors have multiple .cam files for different board types, such as 2-layer or 4-layer boards; if you're offered such a choice, use the 2-layer option for the Pinscape boards.
- Open the appropriate Pinscape .brd file in EAGLE.
- In the EAGLE board editor window, go to the menu and select **File > CAM Processor**.
- In the CAM Processor dialog, go to the menu within the dialog window and select **File > Open > Job**.
- Select the .cam file that you downloaded from your PCB vendor site and click Open.
- Find the "Silk Top" tab and select it. In the Layer list on the right, make sure that layers 200 and 201 are selected. These are the layers containing the snazzy Pinscape logo.

- Also in the "Silk Top" tab, select **either** layer 100, US-Transistor, **or** layer 101, EU-Transistor, according to whether you want to use the US or European transistor option. If you're using the standard parts list, the correct option is layer 100. Layer 101 is only if you want to use the European substitute parts for the small signal transistors; see Chapter 92, European Transistor Substitutions for details.

(This choice doesn't affect anything about the actual electronics, so you can't screw things up too badly here. It only affects the printed orientation guide markings for the transistors. The European transistor substitutes have their legs arranged in the reverse order of the US versions, so they have to be installed "backwards". We offer the two marking options so that you can use the markings that match your choice of parts, to make it easier to insert them the right way when assembling the boards.)

- Click the Process Job button at the bottom.
- Close the dialog and exit EAGLE.
- In the folder containing the .brd file, you'll find a bunch of new files with the same name as the .brd file but with different extensions, including some or all of the following: .GBL, .GPI, .GTL, .GTO, .DRI, .GBO, .GBP, .GBS, .GML, .GTP, .GTS, .TXT. There might be some others as well - the exact set of files you get will depend on your manufacturer. Sorry I can't just give you a list, but it varies by PCB maker! The other obnoxious thing here is that EAGLE just dumps these files into your .brd folder rather than grouping them somewhere else. So the easiest way to identify the files is by "Modified Date" in the Windows Explorer listing. Open the folder in Windows Explorer, switch to the Details view, and select Sort by > Date Modified. You should now see all of the new files grouped together at the top of the listing, all with Modified dates moments ago. These are the Gerber files we've been talking about!
- Once you've identified the collection of new files, create a .ZIP file and add all of the new Gerber files to the ZIP.
- You're done! The .ZIP file is what most PCB vendors will want you to upload.

Note that it's a good idea to read fully through your PCB maker's instructions for submitting orders to make sure there aren't any extra steps they require that we've left out. You should also check to see if your PCB maker accepts .brd files directly: if so, you should be able to skip this whole Gerberfication process and just upload the .brd file.

Checking that your vendor can make the boards properly

If you want to double-check that your chosen vendor has the necessary manufacturing capabilities to make a working set of Pinscape boards, EAGLE provides another automated process that can help. It's known as a "design rules check". Your vendor can provide you with a special file, known as a Design Rules or DRU file, that specifies their manufacturing limits, such as the thinnest copper trace width, minimum spacing between traces, and smallest drill hole. EAGLE can read this file and check it against the board layout, to make sure that the board layout is within the tolerances and limits that the manufacturer specifies. If EAGLE finds any problems, it'll show you warning messages explaining what's wrong. These warnings indicate places where the board design might not turn out correctly in the manufacturing process. Every vendor has their own limits, so these checks have to be made on a vendor-by-vendor basis.

It's up to you whether or not you want to go to the extra trouble of running the design check. I personally would do it when working with a new manufacturer, but

only because I'm meticulous about these things. I don't actually think it's that important to run separate tests for each vendor, because the Pinscape boards are pretty low-tech by modern standards. All of the manufacturers I've looked at have better tolerances than are required for the Pinscape boards. Most people making PCBs today are using very high densities with tiny little surface-mounted parts that are intended to be assembled by robots. The Pinscape boards intentionally use larger "through-hole" parts (the type with leads and pins that get inserted into holes in the board) to make them easier for us hobbyists to assemble by hand. The PCB makers all use processes that accommodate those denser surface-mount boards, so the Pinscape boards shouldn't faze any of them.

If you do want to run the validation process, it's similar to the Gerber generation process described earlier:

- Find the EAGLE .DRU ("design rules") file on your PCB vendor's site. Most vendors include this on their submission instructions page, alongside the .CAM file for generating Gerbers. Download the .DRU file.
- Open the .brd file in EAGLE.
- In the EAGLE board editor window, go to the menu and select **Tools > DRC**. ("DRC" stands for Design Rules Check. Autodesk really knows how to make things intuitive, don't they?)
- In the dialog, click the Load button. Select the .DRU file you downloaded from the PCB vendor and click Open.
- Click the Check button.
- If there are any problems, a dialog will pop up with a list of warnings. If there's no dialog, there are no warnings.

If you do get warnings, you should check each one. You can ask on the forums or contact me if there's anything you're unclear about or that you think might be a serious issue.

Note that you can ignore any "Overlap" errors. Those are due to the inelegant way that we designed some of the circuit traces (they're the way they are due to limitations in EAGLE that I don't know how to work around in a more elegant way), and any that remain in the "Release" versions of the boards will have already been carefully examined and deemed to be intentional. That doesn't stop EAGLE from warning about them, unfortunately, so you just have to know to ignore them when you see them. For each Overlap error, you can click the "Approve" button to tell EAGLE not to warn you about that particular item again in the future.

There might also be some "approved errors" in the boards. You can also ignore these, as "approved" means that I've already reviewed them and determined that they're intentional.

94. Building the Expansion Boards

If you've worked with PCBs before, you should find building the Pinscape boards to be pretty straightforward. The boards can look pretty daunting to build, because there are admittedly a lot of components to install, but all of the components are individually pretty easy to handle. All of them are "through-hole" parts, with wire leads that install through holes in the board. That type of part is easy to solder by hand. (Modern electronics has largely moved to "surface-mount" parts, which are more designed for installation by robots, and can be tricky to solder by hand. I think it's in keeping with our fundamentally retro hobby that these boards use the older style parts that are more human-friendly.)

If this is your first time working with electronics or building printed circuit boards, you might want to browse through our introductory section, A Crash Course in Electronics. The expansive-sounding name notwithstanding, we've tried to keep it focused on the practicalities of building the Pinscape boards, so hopefully it won't waste too much of your time on academic points. Among other things, you'll find help there with reading the markings on the blank circuit boards, so that you can figure out where the parts go, and an introduction to the different component types used in the boards.

The electronics section also some advice on soldering technique (see Chapter 68, Circuit board assembly tips), in case you're looking for help with that. I've found this to be an area that a lot of new cab builders are particularly worried about when embarking on this project because of unhappy past experiences with their soldering irons. There's definitely a little bit of technique needed for successful soldering, but it's not all that hard to learn the basics, so I've tried to offer a few tips that will hopefully make things easier this time around.

Where to begin

Before you start the assembly process, look over the "Optional elements" below and decide if there's anything you want to omit or change in the default configuration. There are some parts you can just omit entirely if you don't need them, and there are variations in the parts you can choose from to slightly change the way some parts of the board work.

When you're ready to proceed, gather the necessary items:

- The blank PCBs - see Chapter 93, Fabricating the Expansion Boards
- The component parts - see Chapter 91, Electronic Parts List
- Soldering tools and other miscellaneous tools - see Chapter 87, Tools

Then just start fitting the components into the boards and soldering them in place. There's no particular order that you have to use to install the components, but I usually like to start with the smaller and flatter components and finish with the taller ones, to avoid having to squeeze into tight spaces between the larger parts.

Component cheat-sheet

Matching the parts to their spots on the circuit board might seem a little intimidating at first, but the "language" of the circuit board markings is designed to make it as straightforward as possible. All of the markings on the circuit board are there to let you see at a glance what goes where. They also tell you how to orient the parts where that matters. (Some parts, like resistors, don't care which direction they go;

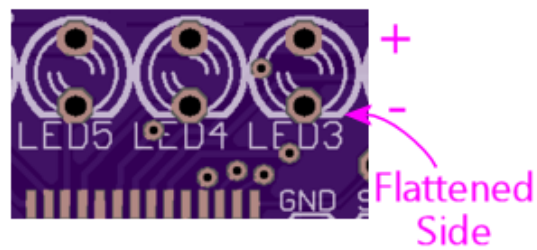
other parts, like transistors and diodes, have to be oriented the right way to work properly.)

There are only about a dozen different types of components on these boards, so you'll see the same "footprints" used repeatedly. Once you learn to recognize the common ones, you'll be able to quickly identify the right parts.

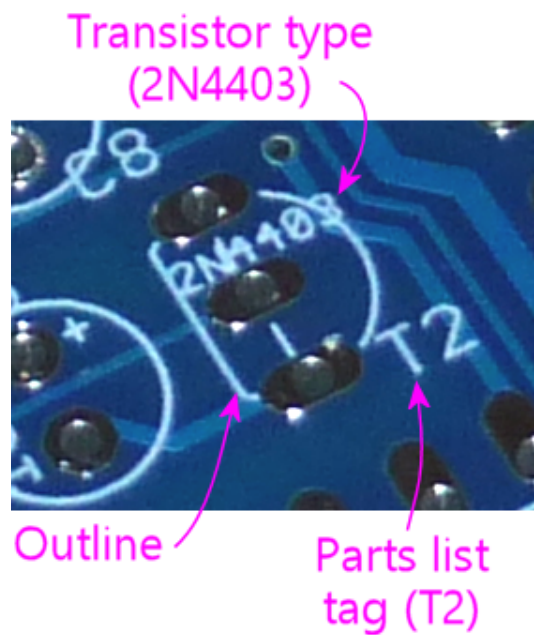
Here's a quick visual reference to the circuit board markings for the various component types. See the linked sections for more details on the component type.

Component	Board Markings	Example
Chapter 72, Resistor	 <p>Parts list tag (R43)</p> <p>Outline</p> <p>Value (82R = 82 Ohms)</p>	
Chapter 73, Capacitor (ceramic disc)	 <p>Parts list tag (C7)</p> <p>Value (100nF)</p> <p>Outline</p>	
Chapter 73, Capacitor (electrolytic)	 <p>Value (100uF)</p> <p>Parts list tag (C11)</p> <p>"+" Marker</p> <p>Outline</p>	
Chapter 74, Diode	 <p>Parts list tag (D1)</p> <p>Outline</p>	

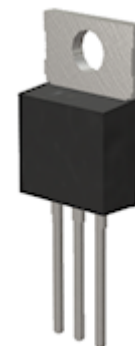
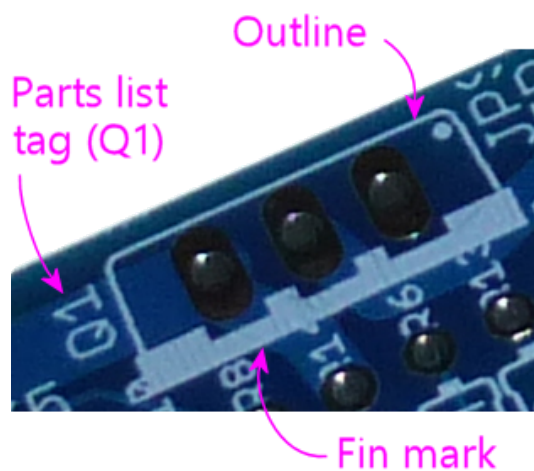
Chapter 75,
LED



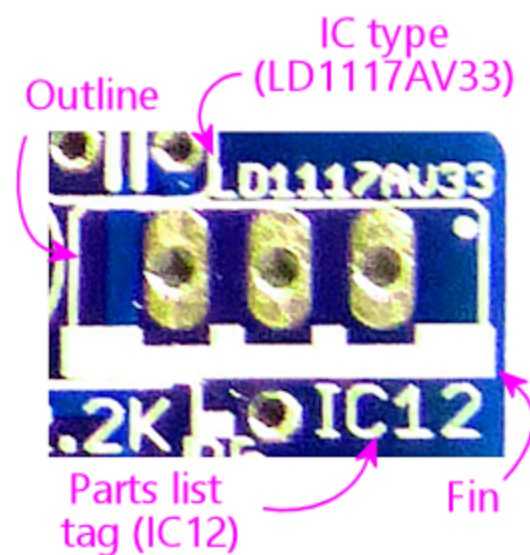
Chapter 77,
Transistor
(TO-92
package)









Chapter 78,
MOSFET
(TO-220
package)



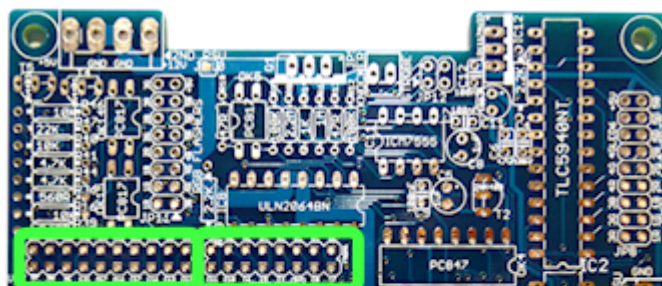
Chapter 79,
LD1117AV33
(IC chip,
TO-220
package)

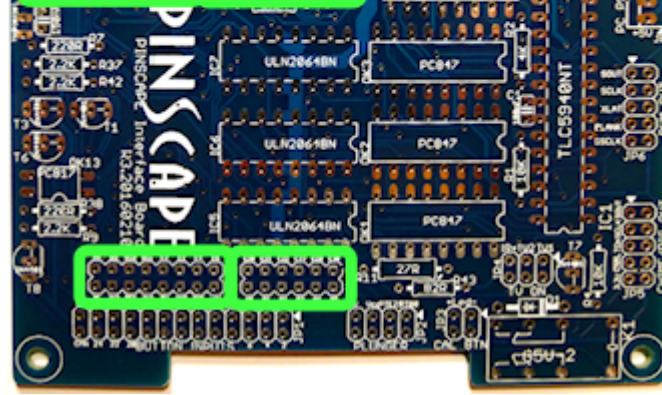


<p>Chapter 79, IC chip</p>	<p>Orientation notch</p> <p>IC type (PC847)</p>  <p>Parts list tag (OK4)</p>	
<p>Chapter 76, Relay</p>	<p>Parts list tag (K1)</p>  <p>Outline</p> <p>Part No (G5V-2)</p>	
<p>Chapter 81, Pin headers</p>	<p>Pin purpose markings (+5V, Vcc, etc)</p> <p>Pin 1 arrow</p>  <p>Descriptive name ("Plunger")</p> <p>Parts list tag (JP2)</p>	

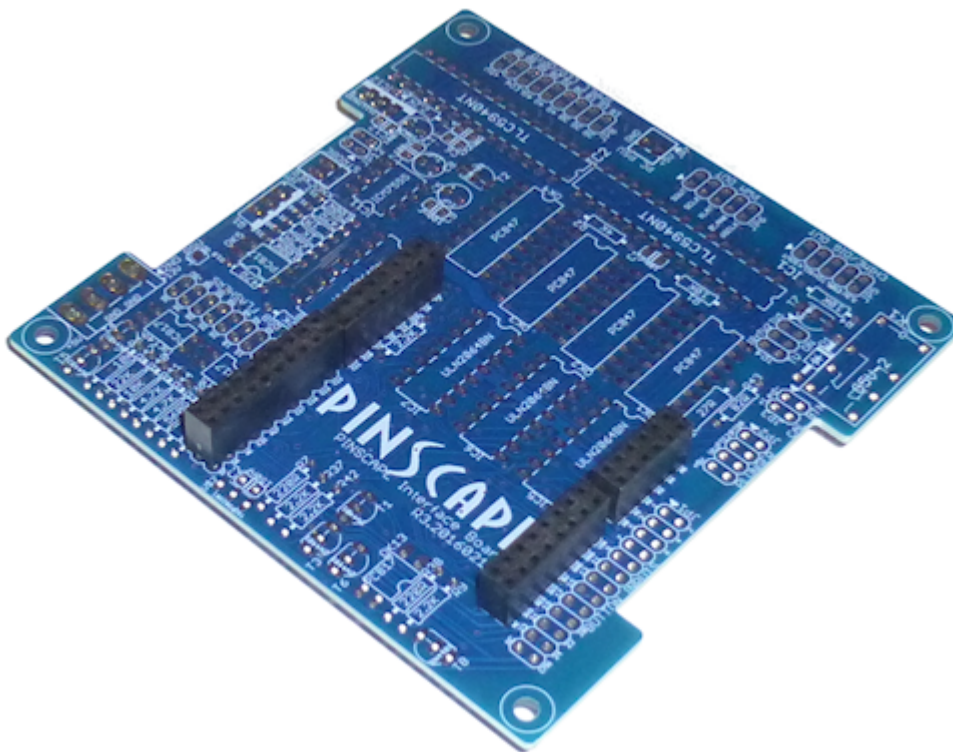
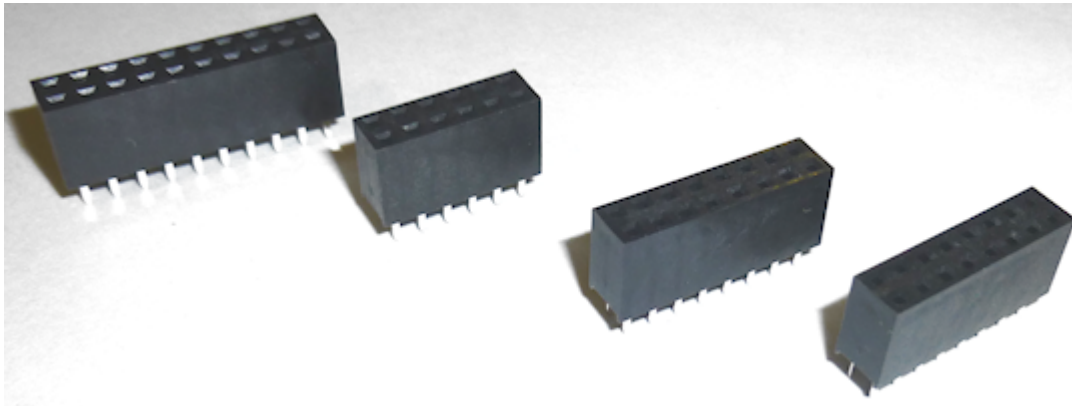
KL25Z sockets

The footprint on the circuit board for the KL25Z consists of four slots that look like ordinary 0.1" pin headers:



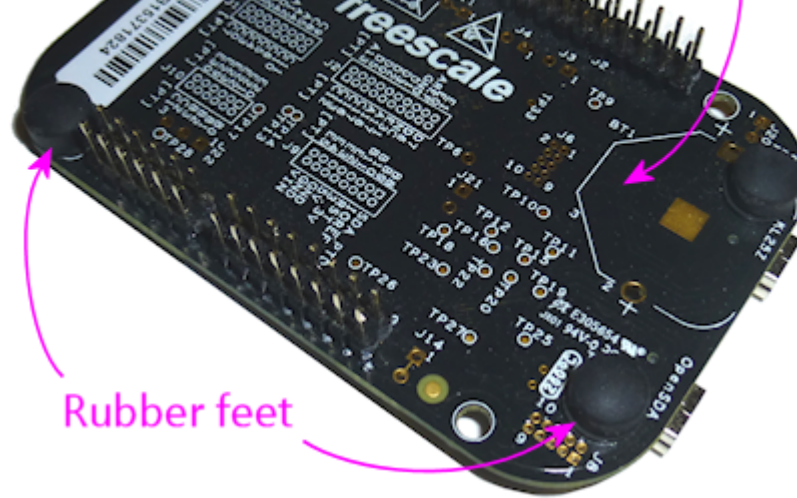


Even though they look like the standard pin header footprints, they're actually intended for 0.1" pin **sockets**, which look like this:



Those mate with the corresponding 0.1" pin headers, which you're supposed to install on the **bottom** of the KL25Z (the side **without** the IC chips).



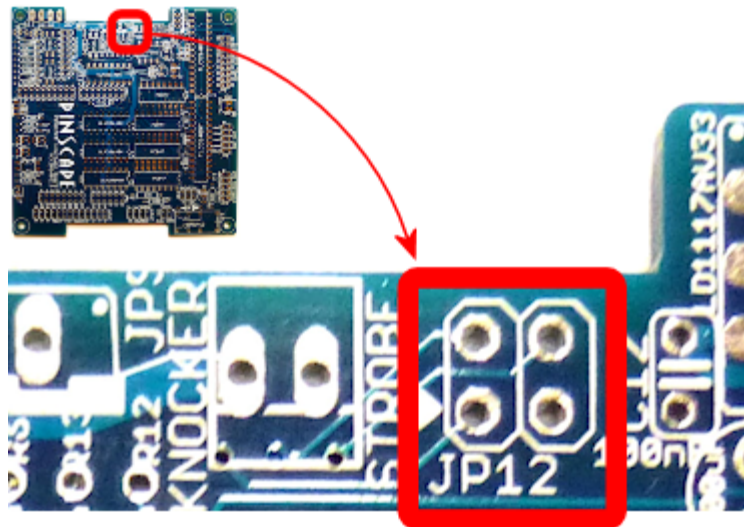


The KL25Z simply plugs into those sockets when you're done assembling the board. Flip the KL25Z top-side-up so that the pins are facing down, and plug the pins into the sockets on the main board.

But don't plug it in just yet! There are a bunch of parts on the circuit board underneath where the KL25Z sits, so you'll need to leave the KL25Z out for now while soldering parts there. For now, just install the sockets on the main board, and the matching pin headers on the KL25Z, and set the KL25Z aside. You can plug in the KL25Z when the rest of the board is done. The KL25Z just plugs in, so it can be inserted and removed at any time; it's not permanently installed.

Special note on JP12 on the main board

This isn't mentioned anywhere in the parts list or on the board itself, but you might want to **omit** the JP12 pin header on the main board. Just leave that slot blank by not installing the pin header.



What this header is for: This header connects back to four unused GPIO ports on the KL25Z. I call it the "expansion port", which I know is a silly port name for an "expansion board", but it fits: this port is for custom additions, if you want to make some custom mods to the firmware to do something with those four free GPIO ports, and similarly for future uses I might come up with some day that I haven't thought of yet.

Why you might want to leave it blank: This header is really confusingly marked (sorry I didn't notice this problem during the design process). See how the word

STROBE is written next to it, with an arrow? A lot of people understandably take this to mean "Hey, guys, the STROBE is over HERE!" But it doesn't mean that at all. The STROBE marking is actually for the next pin header over, that little 2-pin header JP9. The arrow is just the "Pin 1" arrow used on most of the headers, and isn't an "over here" kind of arrow at all.

The confusion is potentially harmful to the KL25Z, because it might trick you into connecting the STROBE device to pin 1 on JP12. That connects directly to a GPIO pin on the KL25Z, so connecting it to a 12V strobe device can destroy the KL25Z.

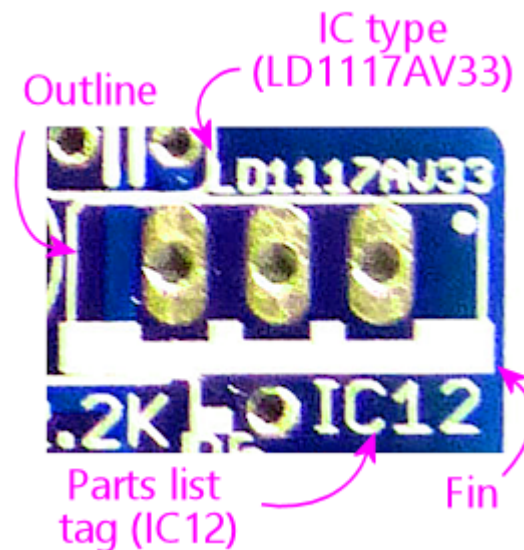
Leaving out the pin header entirely will eliminate that risk.

Why you might want to install it anyway: There actually is one thing that JP12 is good for with the current firmware: you can use it for four extra button inputs. If you're going crazy with the buttons, and you need more than the standard 24 button inputs already on the main board, you can use JP12 to get an extra four.

You can always wait and install the header later if you decide you need it. Or you can just install it up front now that you're aware of which header is which. It's only a risk if you get confused about which pin is for the STROBE, and hopefully that won't be a problem now that you've read through all of this.

Special note on IC12 on main board

Note that the slot for IC12 (a 3.3V regulator IC chip) looks **exactly** like the slot for a MOSFET (the big power transistors we use repeatedly on the power board):

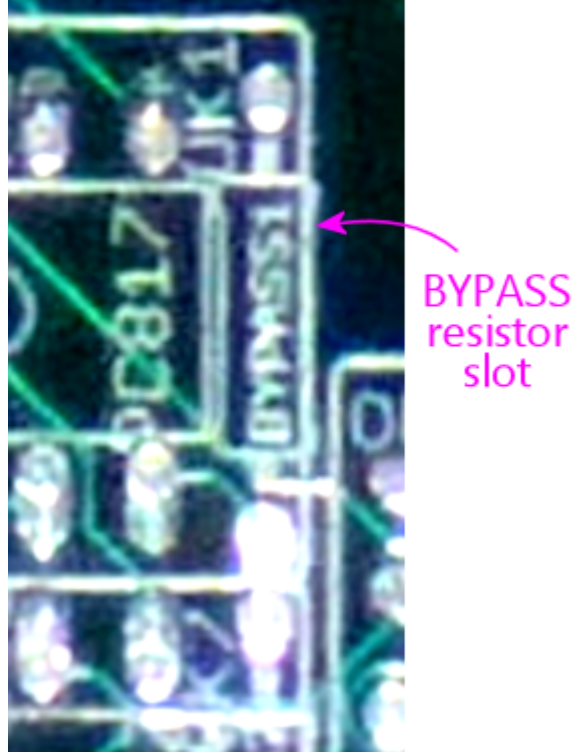


That's because this IC chip is packaged in a TO-220 case, which is exactly the same case type used for all of our MOSFETs.

Installing this part is exactly like installing a MOSFET. The only thing you have to be careful about is making sure you use the right part! Don't install a MOSFET there; be sure to read the label on the part and install the LD1117AV33 regulator chip. You can't tell the difference just looking at the shape of the case - you have to read the tiny text printed on the face of the case.

Special note on BYPASS resistors on the chime board

The chime board has eight resistors labeled **BYPASS**. Normally, you **don't** install anything there. Just ignore this slot and leave it empty. One less part to solder!



The BYPASS slot is there for the "Timer Bypass" option, which lets you skip the timer protection circuits for the replay knocker and/or any of the chime board outputs. See "Bypassing the timers on the chime board" later in this section for details on how to do that.

Optional elements

Some of the components on the boards are optional. You only need to install them if you want to use the features they implement.

Omitting the JP12 pin header on the main board

Read the "Special note about JP12" above for why you might want to omit this pin header.

Configuring the "Small LED" current level

The "Small LED" ports on the main board (on pin header JP8) are "constant current" outputs. This means that they have a built-in current limiter that caps the current at a selected level, which allows you to connect LEDs to these outputs **directly**, without any current-limiting resistors. The function of the current-limiting resistors is replaced by the built-in current limiter, eliminating the need to install the resistors separately.

All of the Small LED ports have the same current limit, but you can adjust what that common limit is, because it's determined by the value of resistor **R5**. If you use the R5 value specified in the parts list, 2.2K, the limit is set to 20mA. You can change this by using a different value for R5.

You can select any value from 10mA to 60mA. To determine the resistor value for a desired current level, use this formula:

$$(\text{Resistance in Ohms}) = 39060 \div (\text{Desired current in milliamps})$$

For your convenience, here are some pre-figured values showing the standard resistor sizes that approximate some selected current levels:

Current	R5 value
10mA	3.9K
20mA	2K
30mA	1.3K
40mA	1K
50mA	750Ω
60mA	680Ω

Don't use a resistor that would set a current above 60mA. Higher currents might stress or even overheat the TLC5940 chip.

Any small-wattage resistor (1/8W or 1/4W) with the chosen Ohms value will work here.

Omitting the TV relay

If you don't need the TV ON feature, you can simply omit all of the following parts:

- D1 (1N4007)
- K1 (G5V-2 relay)
- OK15 (PC817)
- R39 (560R)
- R49 (2.2K)
- T6 (2N4401)

Omitting the TV IR remote features

If you don't need the IR remote control features, you can simply omit all of the following parts:

- C2 (150nF)
- R7 (220R)
- R9 (2.2K)
- R11 (27R)
- T8 (2N4401)
- U\$2 (TSOP38438)

Configuring the IR remote for 1 or 2 emitters

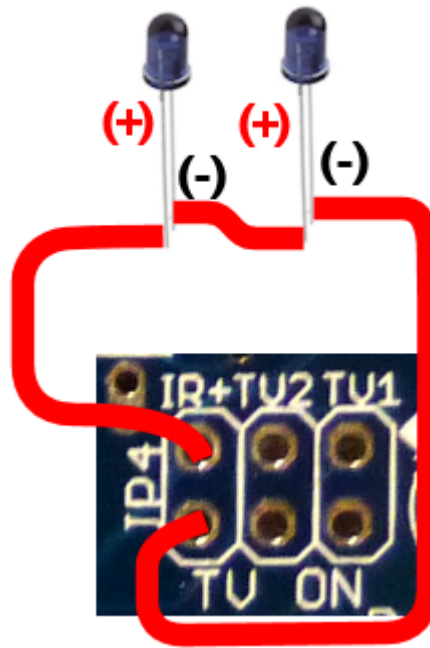
If you're including the IR remote control features, you can build it to power either one or two IR "emitters" (the IR LEDs that transmit the remote control signals).

Why might you want to use two emitters? To control two TVs! If you use two emitters, you can place one near each TV's remote control receiver, to ensure that both TVs get strong signals. Given the way things are packed into a pin cab, it might not be possible to set up a single emitter with a good line of sight to both TVs.

The only change you have to make between one IR emitter vs. two IR emitters is the value for resistor **R11** on the main board:

- For one IR emitter, use a 39Ω , 1/2W resistor for R11
- For two IR emitters, use a 27Ω 1/2W resistor for R11

When connecting two emitters, connect them **in series**. This means that you wire them in a daisy chain:



Recall that the longer leg of an LED is the positive (+) lead. IR LEDs such as these emitters follow the same rule.

Omitting the knocker output

The whole knocker output circuit is optional. To remove it, simply omit all of the following parts:

- C5 (1uF)
- C7 (100nF)
- C8 (1uF)
- C9 (100nF)
- IC11 (ICM7555)
- OK5 (PC817)
- Q1 (MOSFET)
- R6 (1M)
- R8 (2.2M)
- R10 (100K)
- R12 (100K)
- R12 (100K)
- R13 (47R)
- R14 (1K)
- R18 (47R)
- R37 (2.2K)
- T2 (2N4403)

- T3 (2N4401)

The list above includes all of the parts that make up both the knocker's time-limiter circuit and the output control circuit, so if you omit all of those parts, you can skip the section below about bypassing the knocker output timer. The output timer won't be there at all if you omit the parts above, so there's nothing left to bypass!

Bypassing the knocker output timer

The replay knocker output on the main board has a built-in hardware timer that cuts off power to the output if it stays on continuously for more than a couple of seconds. This is designed to protect your replay knocker from software faults on the PC - it's been known to happen that Visual Pinball can crash in the middle of an operation that leaves the knocker coil energized. Leaving a knocker coil on for more than a few seconds can overheat it and burn it up. However, if you like to live dangerously, or if you want to re-purpose this output for a different kind of device that doesn't need the timer protection, it's possible to bypass the cut-off timer and turn this into a general-purpose output with no time limiter.

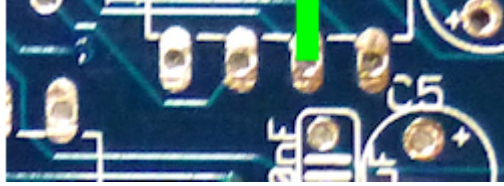
Note: the procedure here omits just the *timer* part of the circuit, leaving the output circuit for the knocker in place. If you want to leave out the *entire* knocker circuit instead, including both the timer *and* the output portion, see "Omitting the knocker output" above.

To bypass the timer circuit and turn this into an un-timed, general-purpose output:

- **Omit** all of the following parts (simply don't install anything in their slots on the board):
 - C5 (1uF)
 - C7 (100nF)
 - C8 (1uF)
 - C9 (100nF)
 - IC11 (ICM7555)
 - R6 (1M)
 - R8 (2.2M)
 - R10 (100K)
 - R12 (100K)
 - T2 (2N4403)
- In place of the 47Ω resistor for R18, install an 82Ω resistor
- Install a jumper wire between pins 3 and 8 of IC11 (remember, we're **not** installing the IC there). "Jumper wire" just means that you can use an ordinary piece of hookup wire.

Jumper wire
pin 3 to pin 8





With these changes, the replay knocker output will turn into an ordinary, general-purpose, digital output with no timer cut-off. "Digital" means that it's purely an on/off port: it doesn't have PWM capabilities, so you can't use it for a device that requires brightness or intensity control.

The power-handling capacity isn't affected by the timer bypass. No changes are needed in the software (either in the Pinscape firmware setup or on the PC), since it's invisible to the software in the first place; the timer (when installed) acts to cut off power at the hardware level without any software involvement.

Bypassing the timers on the chime board

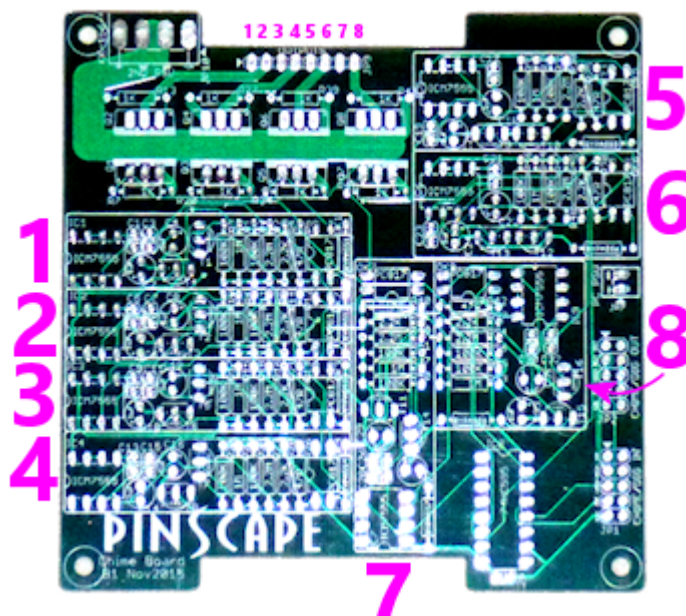
The chime board's outputs are designed to be protected by hardware timers, to prevent a coil from getting stuck "on" in case of a software crash. If you wish, though, you can bypass the timers to turn these outputs into ordinary, general-purpose outputs that you can use for devices that don't need the timer protection. The timers can be installed or bypasses individually for each output, so you create any mix of timer-protected and general-purpose outputs on each board.

Bypassing a timer doesn't have any impact on the software on the KL25Z or on the PC. The timers are completely invisible to the software, since they intervene directly at the power switch level.

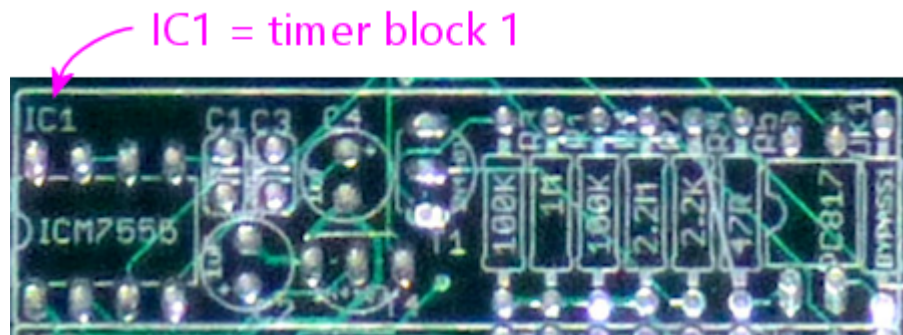
Note that the chimer board outputs are all "digital": they don't have any PWM control capability, so you can't adjust the brightness or intensity on one of these outputs. They can only be fully on or fully off. That's true with or without the timer circuit.

To bypass an individual timer circuit:

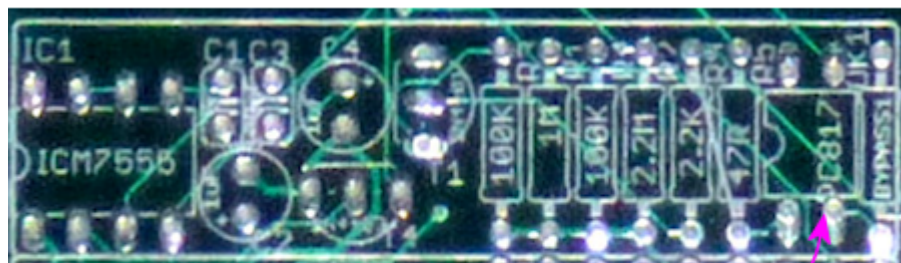
- First, identify the timer block area for the output whose timer you want to bypass. Each output has its own timer, and each timer can be built or bypassed independently of the others. The timer block for a each output is outlined on the board in a white rectangle. The blocks correspond to the outputs as shown below - timer block 1 on the diagram corresponds to output 1, etc.



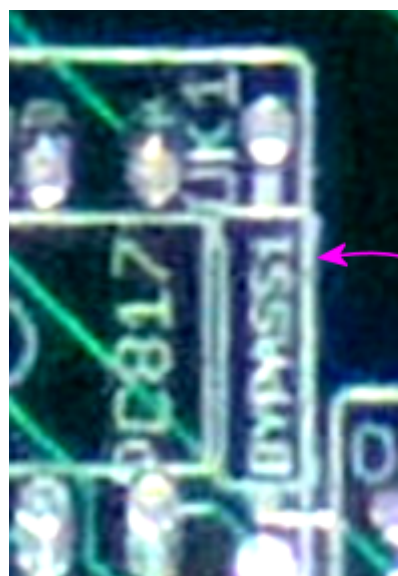
- You can also identify which block connects to which output by looking for the ICM7555 IC chip within the block. These are numbered the same as the outputs: IC1 is in timer block 1 for output 1, IC2 is in timer block 2 for output 2, etc.



- Now we're going to build the "null" timer circuit, without the timer. This means we're going to basically **omit** all of the normal parts.
- The only part from the original timer that you're going to install here is the **PC817** optocoupler chip, also labeled **OK n** where n is the timer block and output number (OK1 for timer block 1 and output 1).

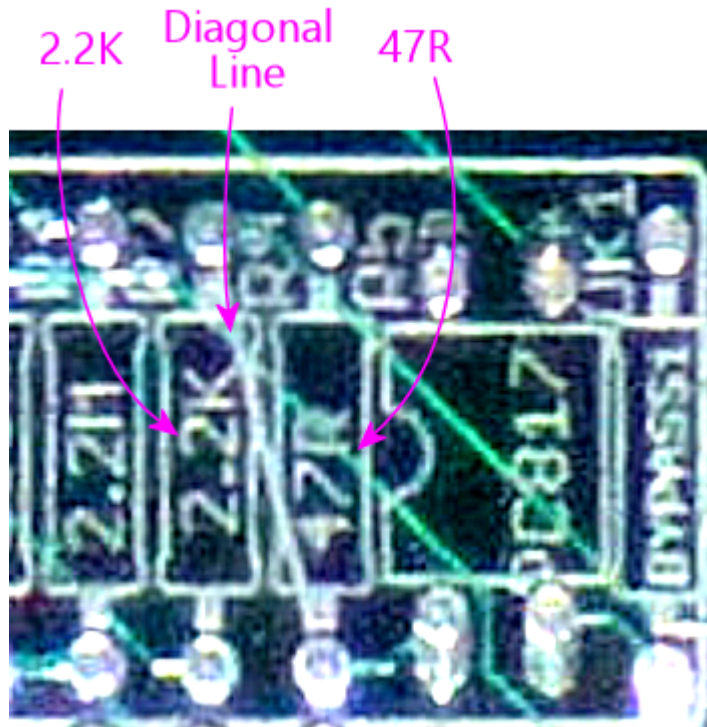


- Now we're going to add two parts within this block that *aren't* part of the normal timer circuit. First, find the BYPASS resistor slot. It's the slot with a resistor footprint, labeled BYPASS n (n is the block/timer number we're working with again).





- Install a **jumper wire** in that slot. In other words, simply install a short piece of hookup wire between the two ends of the resistor footprint.
- The second part is trickier. In the timer block, find the 2.2K resistor and the 47R resistor that are placed right next to each other. Every timer block has a pair like this. And there's a really unusual marking on this pair: a white diagonal line across the pair.



- That diagonal line is where you're going to install the second extra part: a **220Ω resistor**. Install it with one lead going through the solder pad hole at one end of the diagonal line, and the other lead in the solder pad hole at the other end of the diagonal line. (As always for resistors, the orientation of the resistor itself doesn't matter.)

And that's it! You've successfully converted this output into an ordinary output with no timer cut-off.

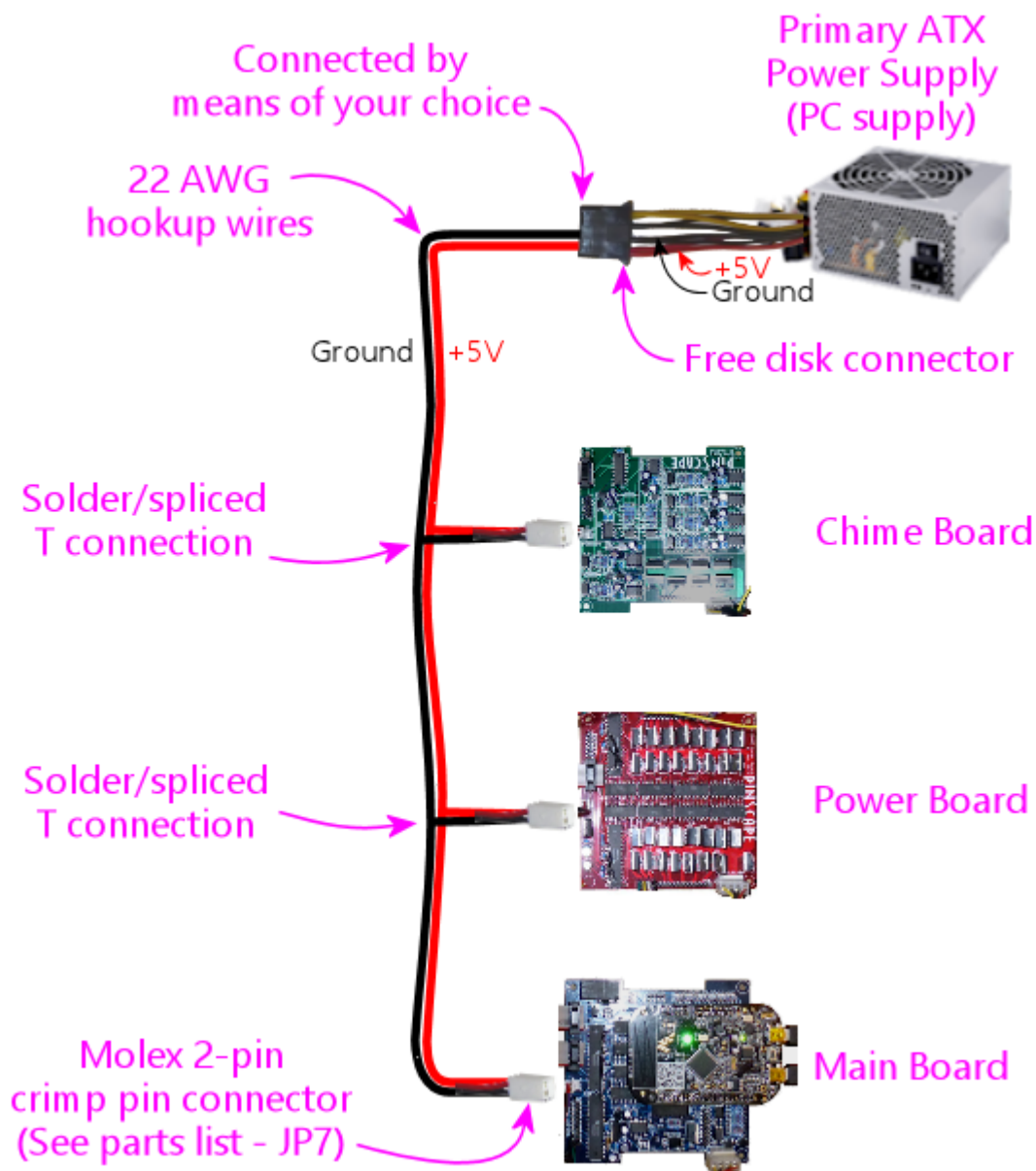
95. Expansion Board Power Cables

Each of the expansion boards has two connectors that have to be plugged into ATX power supplies. These are designed to work with special, custom power cables that you build as part of the expansion boards. This section explains what's needed and how to build the cables.

Connection plan

There are two cables, one for the "primary" ATX power supply, which is the one that powers your pin cab's PC motherboard, and one for the "secondary" ATX supply, which is dedicated to powering feedback devices.

The "primary" cable connects to your PC's power supply at one end, and then runs in a daisy-chain to each expansion board, plugging into the "PC PSU" connector on each board.

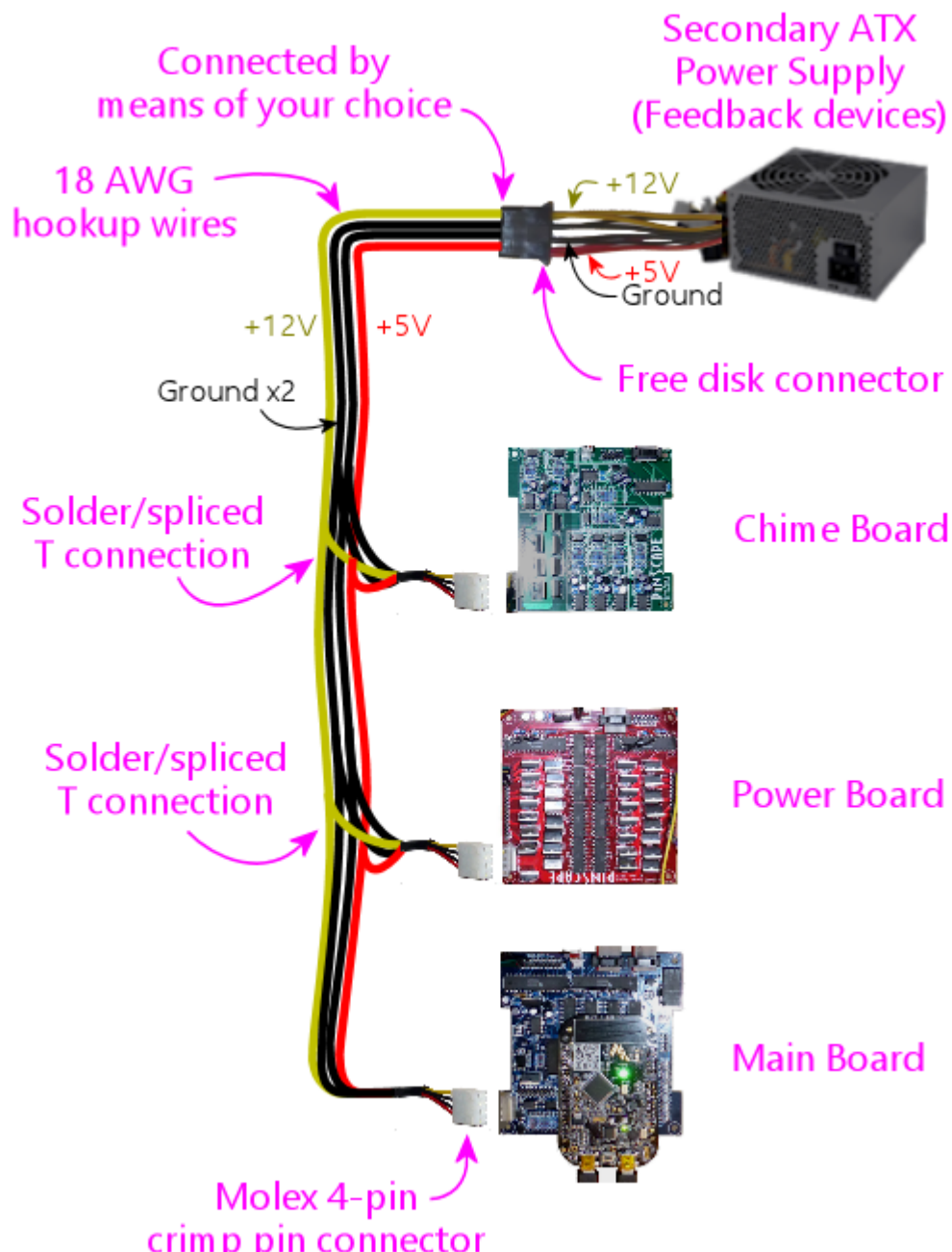


Here are some things to note:

- This is a custom power cable that you're going to construct out of hookup wire and the Molex 2-pin crimp-pin connectors in the parts list

- The connection at the ATX power supply end can be handled in a few different ways, such as using a compatible plug to plug into a free disk connector, using a breakout board, or soldering directly to a disk cable on the power supply; we'll look at the details later in this section
- The primary power connector only needs two wires: the red +5V wire, and the black Ground wire; these connect to the corresponding wires in the ATX disk connector or breakout board
- Each expansion board has its own plug
- The plug for each board is the same type
- Each plug connects back to the main wire with a "T" connection
 - You can solder the connection at each "T"
 - Or you can use a "T-Tap" splice connector for a solderless joint (available at Amazon, Home Depot, etc)

The "secondary" cable, similarly, connects to your secondary PC's power supply at one end, and runs in a daisy-chain to each expansion board, plugging into the "2ND PSU" connector on each board.



Wire

To build these cables, use ordinary hookup wire. Here are the recommended wire types:

- Primary PSU connectors:
 - 22 AWG
 - One black wire
 - One red wire
- Secondary PSU connectors:
 - 18 AWG
 - Two black wires
 - One red wire
 - One yellow wire

Note that the secondary PSU connector uses **two** black wires, which connect to the corresponding black wires in the disk connectors. Be sure to connect both of these even though it seems redundant to use two. The reason we use two is that it doubles the current-carrying capacity. The black wires on these connectors carry the **combined** current from **all** of the feedback devices, so they can be subjected to very high currents at times. It's the same reason the ATX power cables use two black wires - all of the current for all of the different voltages ends up going through these wires.

For each wire:

- Cut one piece long enough to reach from the power supply to the **farthest** expansion board
- Cut one additional piece for each additional expansion board, long enough to reach from the main cable to that board

How to connect to the expansion boards

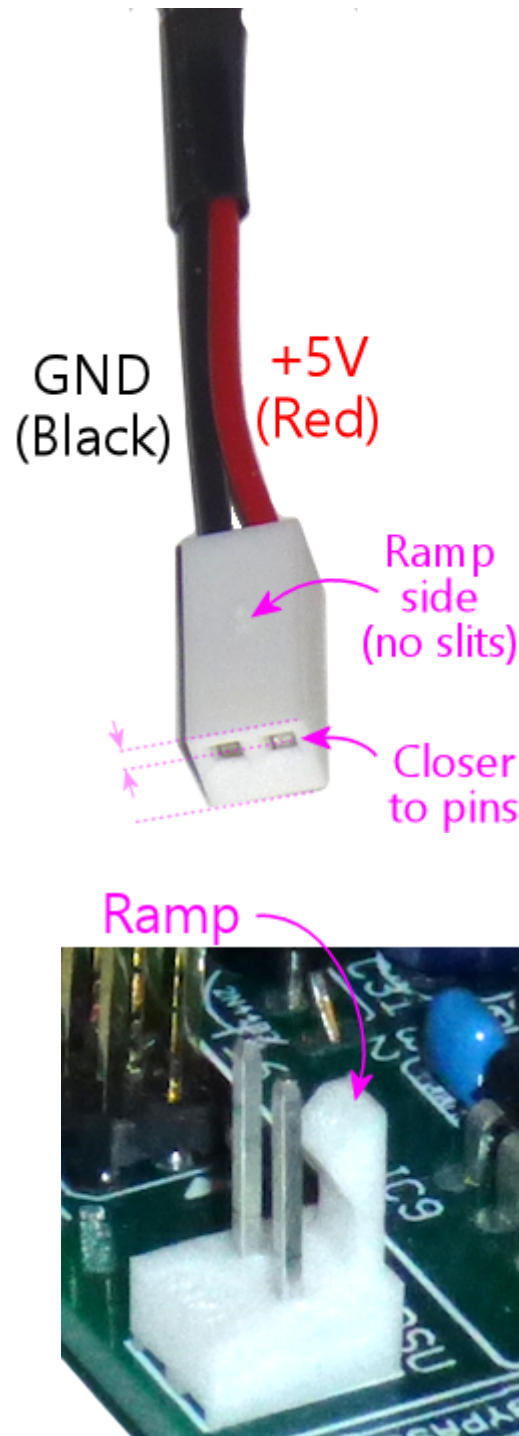
See the Chapter 91, parts list for the necessary connectors. There are two distinct connector types:

- A **two-pin connector** for the **PC PSU** (primary power supply) connections
- A **four-pin connector** for the **2ND PSU** (feedback devices/secondary power supply) connections

These are crimp pin housings, so you have to build them by crimping the pins onto the ends of the wires, and inserting the pins into the housings. Full instructions are in Chapter 82, Crimp Pins.

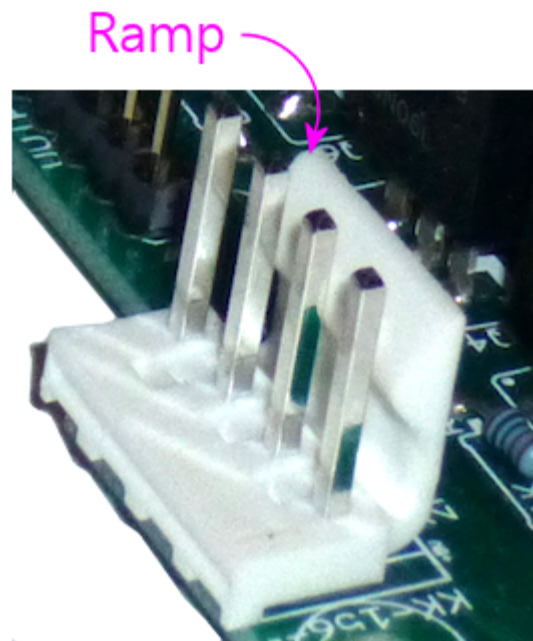
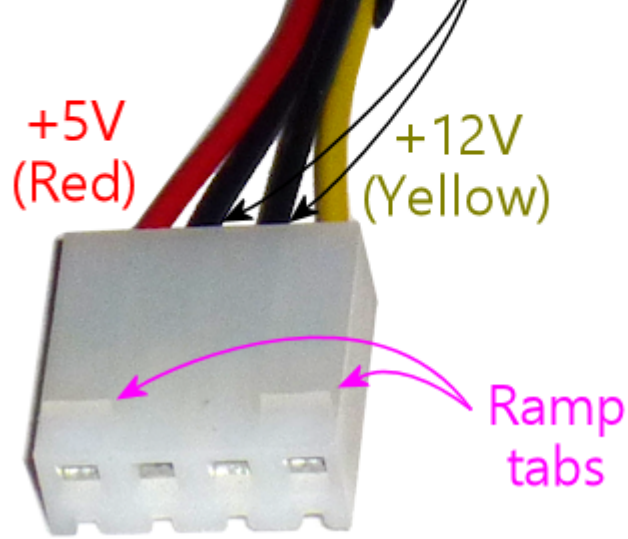
Here's the pin ordering for the two-pin primary power supply connector. To orient it properly, look for the "ramp" side, which is the side that faces the ramp on the circuit board header when you plug it in. (The ramp is the vertical plastic projection on one side of the header, which ensures that you plug the connect in the right way.) The ramp side on this connector isn't marked very clearly, but you can identify it because it's the solid side - the opposite side has narrow slots where the crimp pin lock tabs

fit. You can also tell the difference between the two sides by the distance to the pin openings in the bottom: notice in the picture how the pin openings are closer to the ramp side.



Here's the pin ordering for the four-pin secondary power supply connector. To orient it, look for the "ramp tabs" - the little triangular tabs sticking out on one side of the connector. This is the side that faces the ramp on the circuit board header when you plug it in.





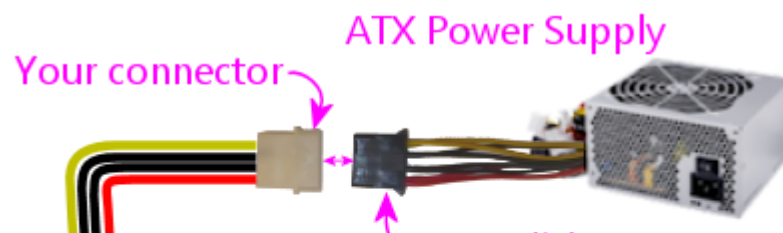
Once the housings are built, you connect them to the Pinscape boards simply by plugging them in to the matching headers. The connectors and headers are keyed so that you can only plug them in one way.

How to connect to the ATX power supplies

One end of each of the special Pinscape power cables connects to one of your ATX power supplies. There are several ways to handle the connection at this end.

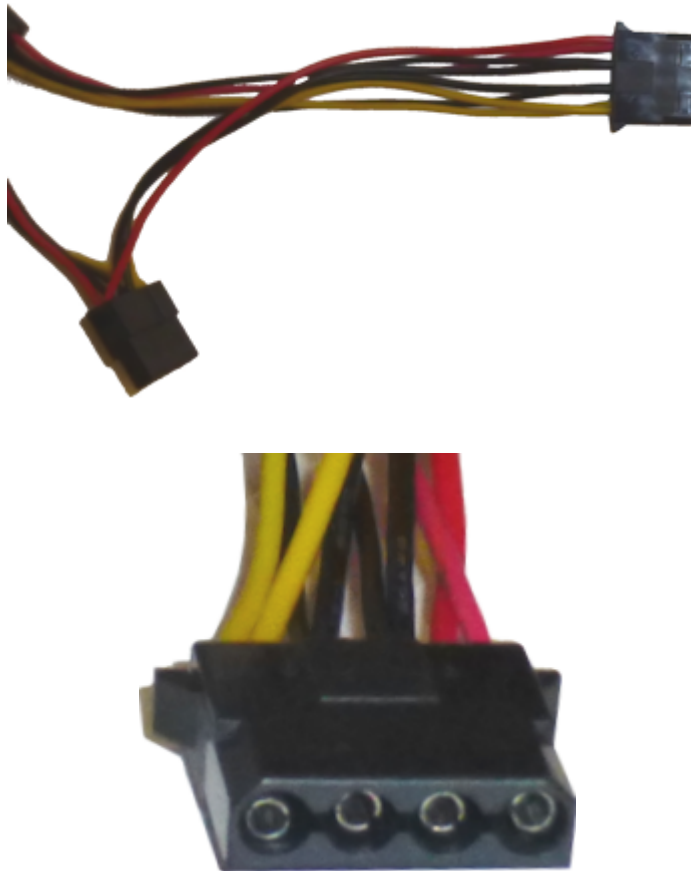
Through a disk connector plug

My preferred way to connect to each ATX power supplies is through a free disk power connector on the power supply. This approach doesn't require any modifications to your power supply; you just plug into it the way you'd plug in a disk drive.



To Pinscape Boards

The disk power connectors are the four-pin connectors that look like this:



This is a standard connector - the brand name is AMP Mate-N-Lok. (It's commonly misidentified as a Molex connector. There is in fact a similar-looking Molex connector, but it has a slightly different plug shape that's not compatible.) The connector that's attached to your ATX power supply is female, so to connect to this, you need the male side:

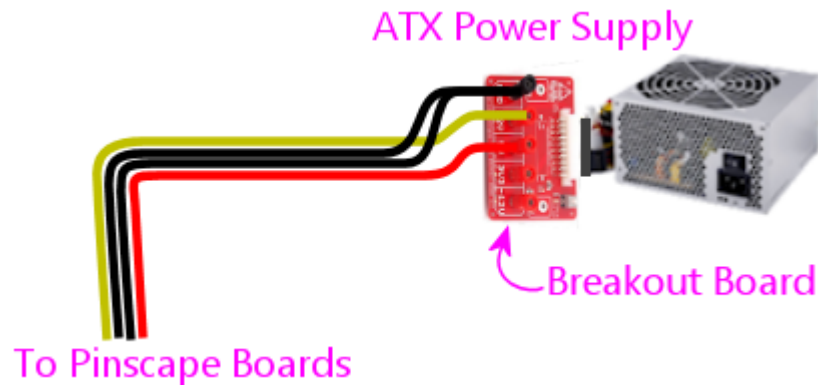
- TE/AMP 1-480426-0 Mate-n-Lok 4-pin housing
- TE/AMP 60620-1 male crimp pins, quantity 4 per connector

These are crimp pin housings, so they require some assembly. See Chapter 82, Crimp Pins for details on how to build this kind of connector.

One slight complication: the AMP crimp pins for this housing are designed for wire gauges from 20AWG to 14AWG. That's fine for the secondary PSU connector cabling, for which I recommend 18AWG hookup wire. But the primary connector cabling can't be thicker than 22AWG because of the crimp pins we use at the expansion board end! You might find that these large AMP crimp pins don't crimp onto the 22AWG wire tightly enough. If it won't stay in place after crimping, you can secure it by soldering the wire to the pins. Which defeats the purpose of the crimp pins (easy of assembly), but works. Alternatively, you can use thicker wire (18AWG, say) at the AMP connector end, and solder those thicker wires to the thinner 22AWG wire needed at the Pinscape end.

Through a breakout board

If you're using a breakout board to access your ATX power supply's 5V and 12V connections, you should use whatever connector type your breakout board uses. For example, if your breakout board uses screw terminals, simply strip 1/4" or so of wire from the end of the custom Pinscape power cable that's going to connect to the breakout board, and screw the ends into the appropriate terminals.

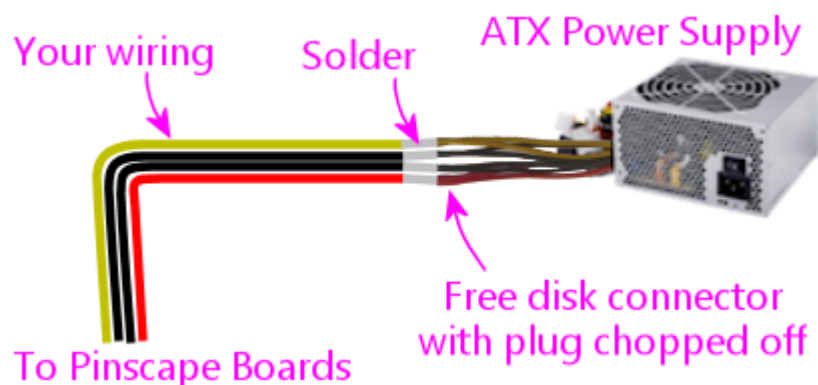


Hard-wired

The simplest way is to solder the Pinscape cabling directly to your ATX power supply's wiring, preferably to one of the disk power connector cables. This isn't my favorite approach, because it permanently modifies the power supply, but it's quicker and easier than messing around with crimp pins, and cheaper than adding a breakout board.

To do this, chop off one of the disk connector plugs on the power supply. That'll leave you with four bare wires you can solder to. Strip 1/4" or so of insulation from the end of each wire. Do the same with the wires in the Pinscape cable. Solder the ends together. Cover the exposed solder joints with shrink-wrap tubing or electrician's tape.

This leaves you with an ATX power supply with one less disk power cable, and a permanently attached Pinscape power cable. You can still plug and unplug the Pinscape ends, so there's no loss of modularity.



Frequently asked questions about the power connectors

I've helped a number of people debug problems that came down to missing power connections, so I wanted to clarify some points about the power wiring. Some of the confusion probably comes from the LedWiz and some of the other arcade devices that have a bunch of connectors that you don't really need, so people probably get used to the idea that the connections on these controllers are mostly for "other

people" with unusual setups. Not in this case! The connectors on these boards are for everyone. So, to answer some questions that have come up several times:

- **Do I really need to connect the 5V *and* 12V wires in the secondary connectors?**

Yes! These power connections actually provide power to the logic chips and transistors on the board. If the supply voltages aren't connected, the logic chips and transistors won't have power, and they simply won't work.

- **Do I really need the *both* black wires in the secondary connector?**

Yes! The secondary connector provides the common ground connection for **all** of your feedback device power connections. When your 3A shaker motor runs, that 3A goes through these wires. When your 3A shaker motor runs at the same time as your 4A gear motor and your 2A fan, the combined 9 Amps from all of those devices goes through these wires. We use two wires because it doubles the current-carrying capacity of one wire. We need that extra capacity because of the high combined current that can result when multiple feedback devices are running at the same time.

- **Do I really need to connect *both* the primary *and* secondary connectors to each board?**

Yes! The two power connectors supply power to separate circuits on each board. The primary PSU connector supplies power to the "logic" circuits that connect to the KL25Z, such as the TLC5940 chips on the main and power boards, and the 555 timer chips on the chime boards. If the primary PSU connector isn't connected, none of the logic circuits receive any power, and nothing works. The secondary PSU connector supplies power to the MOSFETs that carry out the power switching, *and* they provide the ground connections for *all* of the feedback devices' power supply connections. If this connector isn't connected, the MOSFETs can't switch and the feedback devices don't have any connection to ground, so nothing works.

- **Do I really have to use *two separate* ATX power supplies for these boards?**

No. It's possible to run everything off of the single ATX power supply that's powering your PC motherboard. You **do** still have to connect both power cables, though - you'd just connect them both to that single ATX supply, if that's all you have.

But this is one of those cases where you shouldn't do something just because you can. The reason that we take it for granted that you're using two power supplies is that sharing a single power supply with feedback devices places too much load on your single PSU, and creates too much opportunity for electrical interference.

96. Initial Expansion Board Testing

This section gives you some tests you can perform on your newly built expansion boards to make sure they're working. I'd recommend going through these tests with a new board before installing it in your cabinet, since it's a lot easier to isolate and debug problems before everything is connected. The test procedures here are designed to test each subsystem in isolation (as much as possible).

If anything isn't working properly, the next chapter, Chapter 97, Debugging the Expansion Boards, offers some ideas for how to diagnose and fix any problems.

Configure the firmware

Before plugging anything into the expansion boards, you should make sure that your Pinscape firmware is installed, and that you've configured it for the expansion boards. Some of the GPIO pin assignments are different between the standalone configuration and the expansion board configuration, so things won't work properly if you don't set the right system type.

You can do all of the initial software setup and configuration with the KL25Z on its own, not plugged into the expansion boards.

For initial firmware installation, see Chapter 89, KL25Z Software Setup.

To configure the firmware for the expansion boards:

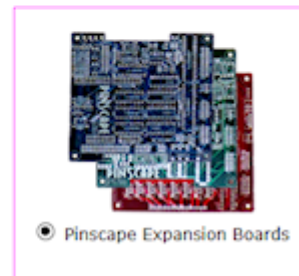
- Launch the Pinscape Config tool
- Click the Settings icon for your device
- Make sure the **System type** (near the top of the page) is set to **Pinscape Expansion Boards**

Settings

Configuration settings for KL25Z CPU ID 0039-004D700F-33024E45

Memory used: 10K bytes of 16K bytes (5,676 bytes free) [Explain](#)

System type. Are you using this KL25Z on its own, or with a set of expansion boards? [Help](#)



- Set the boxes below that to reflect the correct number of power boards and chime boards you're going to connect

Tools and supplies for testing

For the initial testing, it's really helpful to have some items available:

- An ATX power supply. You can use the power supply that you're using with your pin cab motherboard if it's convenient, or you can use a separate one.

If you're using a separate power supply, make sure you've hacked it so that it powers up without a PC motherboard attached. If you're using a "breakout board" to get at its power connections, that should do the trick. If you're using your own connectors, you have to bridge a pair of pins on the motherboard connector to tell the PSU to power on. See "Overriding the soft power circuit" in Chapter 45, Power Supplies for Feedback for the procedure.

For testing purposes, it's okay to just use a single ATX power supply. As we've said before, it's best to use two separate ATX supplies for your full cab setup - one for the PC, a separate one for the feedback devices. And if you have both power supplies conveniently available, you might as well test with the full setup. But it's also okay to use a single power supply for now, since we can do all of the testing with low-power devices.

- The special expansion board power cables, to connect between the ATX power supply and the Pinscape board. Two cables are required - one for the primary power supply connection, one for the secondary power supply connection. The two cables use different connectors (the primary uses a two-pin connector, the secondary uses a four-pin connector). You need both cables, even for testing purposes. See Chapter 95, Expansion Board Power Cables for instructions on assembling the cables.
- Some "Dupont" cables, for making temporary test connections to the 0.1" pin headers. You can find these on eBay by searching for "Dupont cable", and you can get them from any of the hobby robotics Web stores (Pololu, SparkFun, Adafruit). These are simply hookup wires with individual 0.1" pin header male pins/female sockets attached to the ends. They let you safely plug a wire into any pin on any of the headers to test the connection. It's convenient to have a full set of the different gender combinations - male-to-male, female-to-female, and male-to-female.

If you buy a set of these on eBay, they'll often come as a sort of "ribbon cable", with all of the wires joined together in a flat bundle. For testing purposes, we'll just need a few individual wires, with one pin on each end. You can peel individual wires (or groups of wires) off of the ribbon cable bundles as needed simply by zipping them along the seams between the wires. The insulation should tear easily at the seams. You can pull off individual wires as needed - no need to separate them all up front.

- Alligator clip cables are also handy, for making temporary test connections to everything else, apart from the 0.1" pin connectors. These aren't a good substitute for the Dupont connectors, though, since it's hard to use them with pin headers; it's too hard to keep the bare metal of the jaws from coming into contact with the pins adjacent to the one you're trying to connect to.
- A small "bubble" LED - the type that runs on 5V or less and draws about 20mA.
- A 220 Ω resistor to use with the LED. (220 Ω should work with any 20mA LED and a 5V supply, but you can find the exact value for your particular resistor using the calculator in Chapter 52, LED Resistors.)

We're going to assume you have these items on hand, but feel free to improvise or substitute similar items as you see fit.

Cautions about connecting power

As a general rule, make sure the power supplies are OFF whenever you plug in or unplug the expansion board power cables.

Similarly, make sure that all power is OFF and that both USB cables to the KL25Z are unplugged when inserting the KL25Z into the expansion board sockets or removing it.

Static electricity precautions

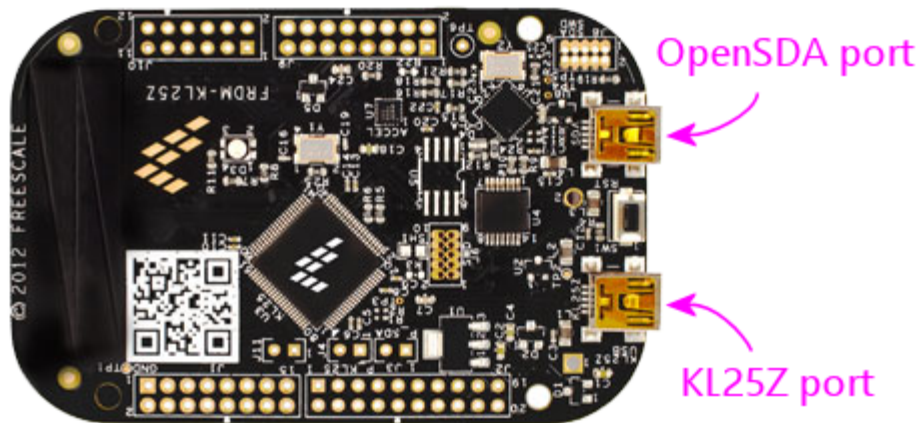
It's a good idea to ground yourself from time to time while working with the bare boards. Touching the metal case of a plugged-in ATX power supply is a good way to do that, or the metal back plane of a PC. See Chapter 67, Static Electricity Precautions.

Connect the KL25Z and the power supplies

Nothing should be connected the main board at this point - no power cables, no cables for the button or plunger inputs, no feedback outputs, and none of the connections to the other boards. It should just be completely on its own.

Unplug the KL25Z from USB, and seat it in the sockets on the main board.

Plug in the USB cable(s) between the KL25Z and the PC. You can connect both ports or just the "KL25Z port":



Plug the "PC PSU" and "2ND PSU" cables into the main board. Again, the power supply should be OFF while doing this.

Remember that **both** power connectors are always required. If you're only using one ATX power supply for the testing phase, you still need both cables - just plug them both into the single ATX supply.

Turn on the power

Okay, we're all set up to try applying power. If the PC isn't already booted up, do so now.

Go ahead and turn on the power supply or supplies. Be ready to cut power quickly if anything seems wrong.

Pay attention to any burning odors coming from the board, and check by touch to make sure that nothing is getting hot. If there's any sign of overheating, cut power immediately (by turning off the power supplies - as always, **don't** unplug the

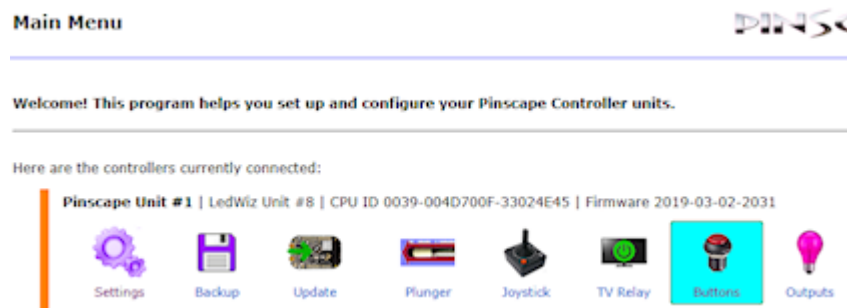
expansion board power connectors while the power supplies are on).

If nothing seems to be overheating, check that the KL25Z has connected to Windows as usual. It should be showing up in the Pinscape Config Tool's list of attached controllers in the main window.

Test button inputs

If the board passes the initial power-on test, the first actual function I'd test is the button inputs, since they're really easy to test.

In the Pinscape Config Tool, click on the Buttons icon for your device.



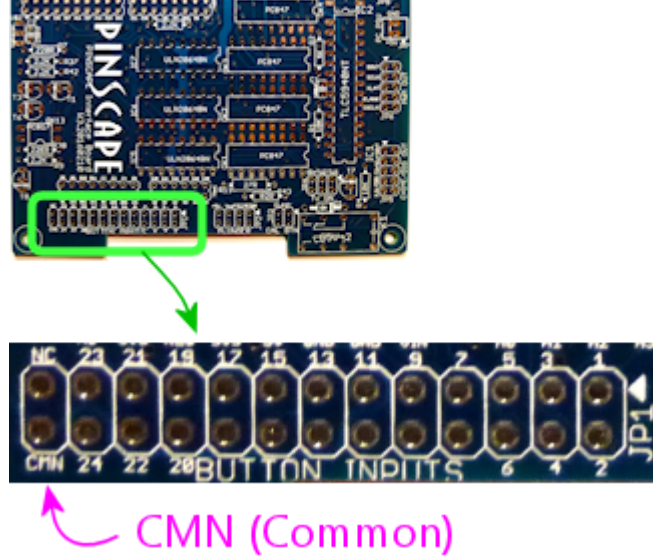
That'll bring up the Button Tester window. This is the most direct way to test buttons, because it shows you the status of the physical GPIO pins. The firmware has to do some work to turn those hardware pin states into keyboard key presses and joystick button events, so other ways of testing are less direct, and any problems could reflect software or setup issues rather than hardware issues. The Button Tester lets us look at the hardware more or less directly to check the basic wiring.

The first thing to check is that all of the buttons are showing their Pin Status as **OFF**. A button input that isn't connected counts as OFF, so assuming you haven't already connected any of the button pins to anything else, an ON indication indicates a problem. Specifically, it means that the wiring for that button is shorted to the KL25Z "ground" connection.

Tip: If button #6 is stuck ON: This is a frequent setup problem that lots of people run into. If button #6 is stuck ON, and the others all look okay, you probably have the System Type still set to "Standalone KL25Z" in the settings. Button #6 has different GPIO pin mappings in the Standalone vs Expansion Board configurations, because the pin originally used in the standalone setup had to be moved to a different function for the expansion boards. This commonly manifests as button #6 being stuck ON, because the software is monitoring that re-deployed pin instead of the pin on the Button Inputs header. To fix this, all you usually have to do is go back to the Settings and make sure System Type is set to Pinscape Expansion Boards.

Next, get out one of the Dupont cables we mentioned earlier. Use a female-to-female type - one with a 0.1" socket at each end. Find the Button Inputs header (JP1), and find the pin on that header labeled CMN (Common). If you orient the board so that the Button Inputs header is at the bottom, the CMN pin is at the bottom left of the header.





Attach one end of the Dupont cable to the CMN pin.

The button inputs work by connecting the individual button pins to the CMN pin. So to test a given button pin, all you have to do is plug the other end of the Dupont cable into the pin you want to test.

Go through the pins one by one. Connect the free end of the Dupont cable to the button pin, and observe the on-screen status in the Button Tester window. The status for each pin should change to **ON** as long as the cable is plugged in to that pin.

Debugging tips: The button inputs are really simple - they're just direct wiring connections from the Button Input header pins to the KL25Z GPIO pins. So there's really nothing that can go wrong with these other than bad solder joints.

The first thing I always try if nothing is working is to swap in a different set of Dupont cables. The cheap ones sometimes have bad connections at one or the other end. I've had this happen with Dupont cables and alligator clip cables multiple times - it's such a huge waste of time when it turns out your test equipment is the problem, and not the board you're trying to test! If you find a bad one, fix it or throw it out immediately so you don't keep stumbling over this red herring on future tests.

Barring bad cables, the most likely problem if none of the buttons are working is the CMN (Common) connection. All of the buttons have to go through that, so it's the prime suspect when the whole thing isn't working. Check the solder joint on the bottom of the board for the CMN pin.

If some buttons are working and some aren't, you know the CMN pin is good. The problem must be in the individual non-working button pins. Check the solder joint for the non-working pin. If that looks good, trace it back to the corresponding pin on the KL25Z connector. Check the solder joints on the expansion board (for the KL25Z socket at that pin), **and** check the solder joint on the KL25Z itself for the corresponding pin.

It's also a good idea to sanity-check the software configuration to make sure the pin you're tracing is the same one the software is using. In the Settings page, in the System Type section, there's a checkbox for "Show KL25Z pin assignments". Check that box, then scroll down to the Buttons list. Each button port will be listed with the KL25Z GPIO port that it's assigned to - "PTC2", "PTB3", etc. Make sure that the connection for the pin you're trying to debug actually does trace back to the port listed in the settings.

Test the TV ON relay

If you installed the TV ON relay, it's easy to test that the relay itself is working:

- Return to the main Config Tool window
- Click the **TV Relay** button for the device
- In the TV ON Tester window, click the **Pulse Relay** button
- You should hear a couple of soft clicks from the relay on the board as it switches on momentarily and then back off
- You can also switch it on indefinitely by clicking the **Relay On** button; click the button again to switch it off

Check the power status reading

Still in the TV ON Tester window, check the **Status**. It should read **PSU2 Power is on**.

If you're using two separate power supplies for these tests, you can test the power sensing circuit further by switching OFF the secondary (feedback device) power supply. (Don't unplug it from the expansion boards - just turn off the power supply itself with its on/off switch.) You should see the status change to **PSU2 Power is off** when you turn off the supply. When you turn it back on, you should see it change to **TV ON delay timer is running** for a few seconds, then switch to **PSU2 Power is on**. You might see it briefly flash **Pulsing TV ON relay** just before that last step, and you should again hear the TV relay click on and off briefly.

Test the Small LED outputs

This is another easy set of functions to test. For this, you need one a small (20mA) LED, and a few Dupont cables and/or alligator clip cables.

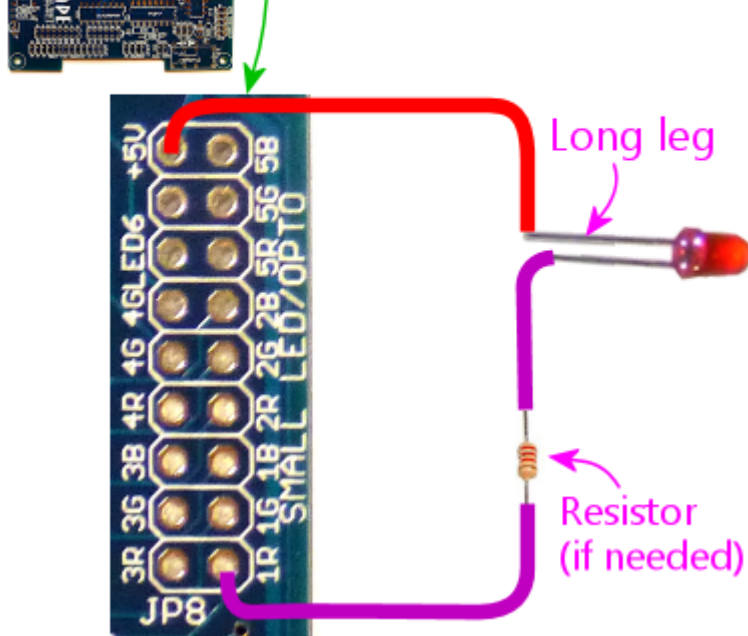
Do this test with the power on. There's no need to shut down before making the LED connections for this test.

You *might* also need the 220 Ω resistor, but probably not. Remember the option to set a custom current regulator level for the Small LED outputs, in Chapter 94, Building the Expansion Boards? If you followed the procedure there to set a custom current regulation level above 20mA, you should use a 220 Ω resistor with this LED. You don't need the resistor now if you used the default current regulator resistor when you built the board. If you don't know what I'm talking about, you probably used the default one, so you don't need the extra resistor now.

Set it up like this:

- Connect the long leg of the LED to +5V on the Small LED header
- Connect the short leg of the LED to the pin labeled "1R" on the Small LED header
- If you need the extra resistor (see above), put it between the short leg of the LED and the "1R" pin

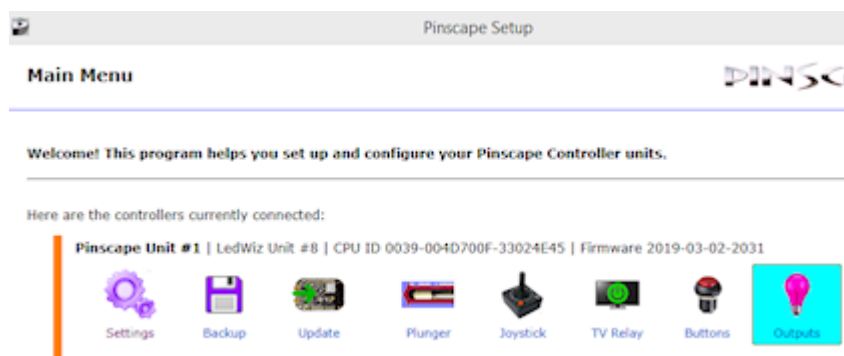




Use whatever combination of Dupont cables and alligator clip cables works for you. I find it easiest to plug into the pin headers with the Dupont cables, and then use alligator clip cables to connect everything else.

Once you have that set up, bring up the Config Tool's output tester:

- Launch the Pinscape Config Tool
- Click the Outputs icon for the device



The Output Port Tester window shows a list of all of your configured output ports. Each port shows the following information:

- The first column, **Port No.** (**Output #** in prior versions of the Config Tool), corresponds to the **Port No.** in the output port list on the settings page.

All of the ports are listed in exactly the same order as on the settings page.

If you're trying to puzzle out how a port in the Settings page list is related to a port in the Output Port Tester list, it's super easy: the first row in the Settings list is the first row in the Output Port Tester list, the second row is the second row, etc. Just look for the port number in the first column of each table.

- The **Pin** column shows the GPIO pin or IC chip port controlling this output.

Most of the expansion board outputs are handled by TLC5940 chips, so these will mostly say things like "TLC5940 #3 OUT7", which means that this is port 7 on TLC5940 chip #3. The TLC5940 chips are numbered in sequential order of attachment; the two on the main board are #1 and #2, the two on the first

attached power board are #3 and #4, and subsequent power boards on the daisy chain have sequentially higher numbers.

Note that the Pin number is **not** related to the output port number in any way. The Pin number tells you the location of the physical circuit controlling the port. The output port number is just an abstract label that DOF uses to refer to the port. It can be confusing to see "OUT6" in the Pin column and think that has something to do with "Port 6" in the settings list or the DOF list. It doesn't - "OUT6" in the Pin list is just the physical pin location on the chip.

- The **Setting** column is where you can test the output. This column shows a slider control or an on/off button for each port. Ports that have PWM (brightness) control get sliders; non-PWM ports just get on/off buttons.

On the sliders, 0 is all the way off and 255 is all the way on. Values in between are proportional intensity/brightness levels, so 128 is about half of full brightness.

Now that you know everything about the Output Port Tester, let's go ahead and test the port. All of the Small LED outputs are controlled by TLC5940 #2, so find the port listed in the Pin column as **TLC5940 #2 OUT0**. If you didn't move any ports around in the settings, this should be **Port No. 50**. Turn the slider all the way up. The LED you wired above should light up to full brightness. You should be able to adjust the brightness by adjusting the slider.

To test the second port, simply move the connector from "1R" to the next port, "1G", and use the corresponding slider control to activate the port. Repeat for each port. The ports are wired to the TLC5940 #2 pins in sequential order:

Pin on JP8	Pin name in Output Tester
1R	TLC5940 #2 OUT0
1G	TLC5940 #2 OUT1
1B	TLC5940 #2 OUT2
2R	TLC5940 #2 OUT3
2G	TLC5940 #2 OUT4
2B	TLC5940 #2 OUT5
3R	TLC5940 #2 OUT6
3G	TLC5940 #2 OUT7
3B	TLC5940 #2 OUT8
4R	TLC5940 #2 OUT9
4G	TLC5940 #2 OUT10
4B	TLC5940 #2 OUT11
5R	TLC5940 #2 OUT12
5G	TLC5940 #2 OUT13

5B	TLC5940 #2 OUT14
LED6	TLC5940 #2 OUT15

Debugging tips: If **none** of the ports are working, make sure that you've got the LED installed in the right direction. It won't light up if installed backwards.

Also try different test cables, in case the ones you're using have bad connectors. I've had this happen more than once!

Using a voltmeter set to Volts DC (VDC), connect the red probe from your meter to +5V pin on JP8, and the black probe to one of the JP6 PWM OUT pins closest to the edge of the board. (Those are all Ground pins.) This should read 5V. If it doesn't, the problem is in the wiring to the +5V pin.

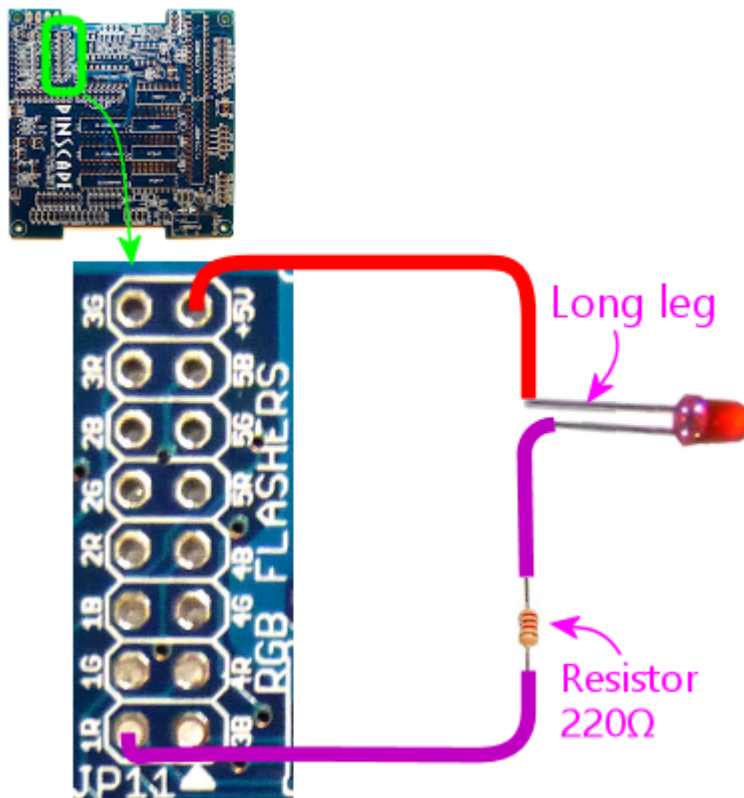
See Chapter 97, Debugging the Expansion Boards for further tests.

Test the RGB Flasher outputs

Testing the flasher outputs is very much like testing the Small LED outputs. For this test, though, you'll definitely need the 220Ω resistor, because the Flasher outputs don't have any built-in current regulation.

The wiring is very much the same as for the Small LED tests:

- Connect the long leg of the LED to the **+5V** pin on **JP11** (RGB Flashers)
- Connect the short leg of the LED to one end of the 220Ω resistor
- Connect the other end of the resistor to pin **1R** on JP11



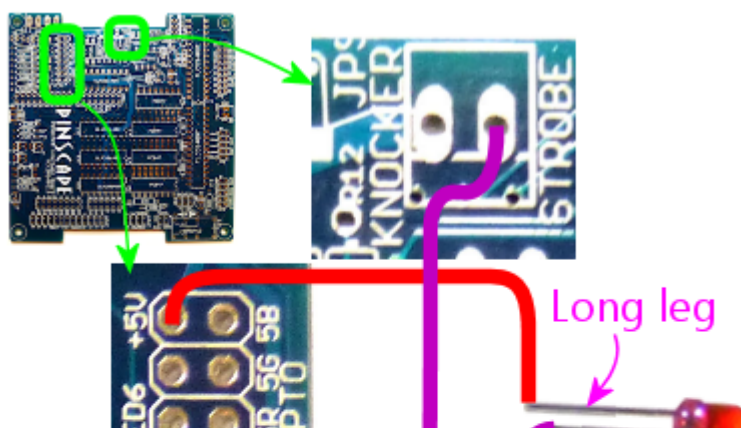
The flasher ports are normally assigned to port numbers 1 to 15. They correspond to the pins on TLC5940 chip #1 as follows:

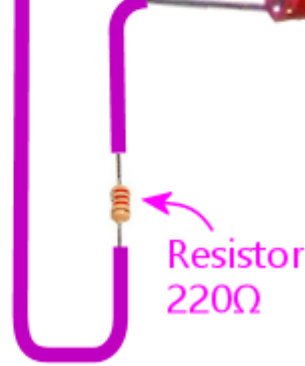
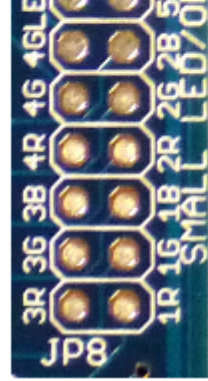
Pin on JP8	Pin name in Output Tester
1R	TLC5940 #1 OUT0
1G	TLC5940 #1 OUT1
1B	TLC5940 #1 OUT2
2R	TLC5940 #1 OUT3
2G	TLC5940 #1 OUT4
2B	TLC5940 #1 OUT5
3R	TLC5940 #1 OUT6
3G	TLC5940 #1 OUT7
3B	TLC5940 #1 OUT8
4R	TLC5940 #1 OUT9
4G	TLC5940 #1 OUT10
4B	TLC5940 #1 OUT11
5R	TLC5940 #1 OUT12
5G	TLC5940 #1 OUT13
5B	TLC5940 #1 OUT14
Strobe	TLC5940 #1 OUT15

Note that the last row is different from all of the others: it's the Strobe output, which we'll come to next.

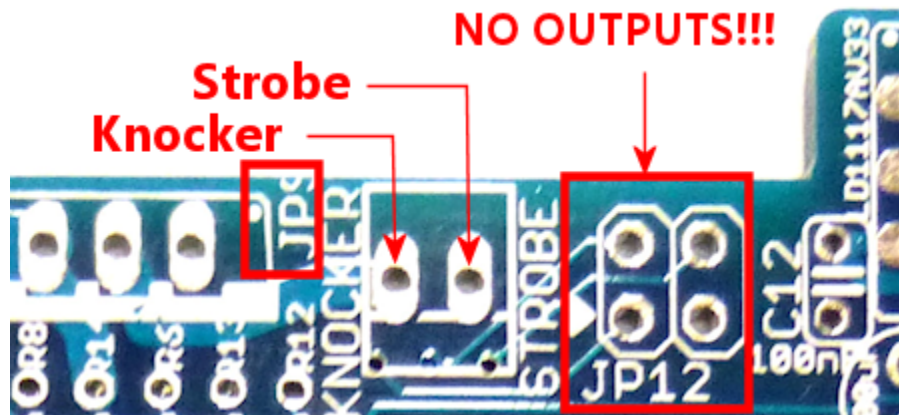
Test the Strobe output

The Strobe output is wired to the same chip that handles all of the RGB Flasher outputs, so you can test it almost exactly like the flasher outputs. It's just on a different pin header. So continue with the same test setup used for the RGB Flashers above, but move the "-" end of the connection over to the Strobe pin on JP9. And in the Output Port Tester window, use the port assigned to TLC5940 #1 OUT15, usually port 16.



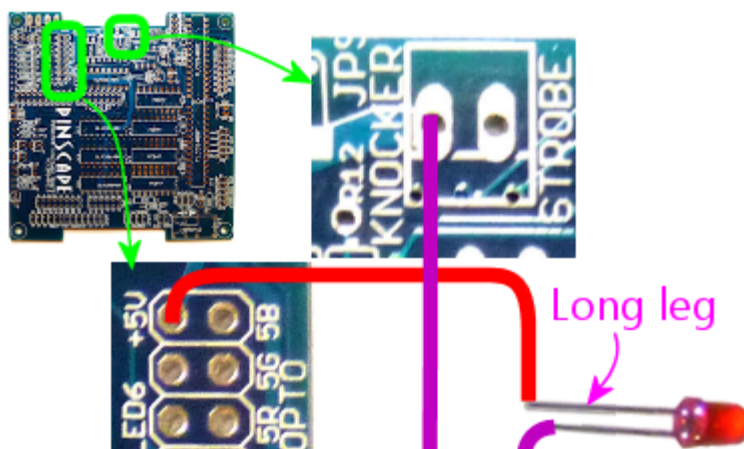


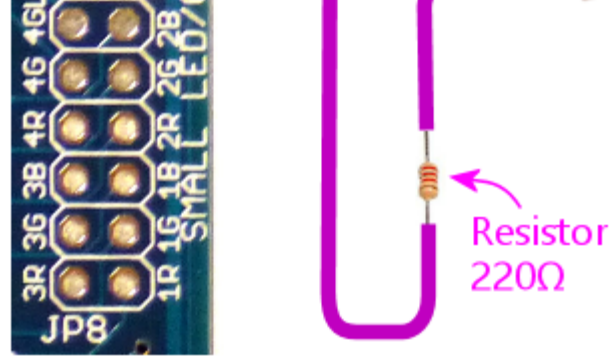
Warning! The Strobe connector is on that little **2-pin** header labeled **JP9**. There's a really confusing marking on the board (my fault for not noticing it in the design process) that makes a lot of people think that the Strobe output is the next one over, on that 4-pin header labeled JP12 - the thing that makes it look that way is that the "pin 1" arrow for JP12 is so close to the word "STROBE" printed on the board that it looks like the two are supposed to be connected, like it's saying "Hey guys, strobe is over here." Sorry for the misleading marking, but that arrow is just the standard "Pin 1" arrow that I use for all of the headers - it's not connected to the word STROBE in any way. The 4-pin JP12 header just connects into four unused GPIO ports on the KL25Z, for custom uses or future software updates. **Never connect a strobe or any other output device to JP12.**



Test the knocker output

You can keep the same test rig we've been using so far, and move the "-" side over to the KNOCKER pin on JP9. That's the one right next to the Strobe pin.





This one has a different appearance for its Pin assignment in the Output Port Tester window: the Pin column for the knocker will show **PCT8 (Digital)**. "Digital" means that it's a non-PWM output, with on/off control only (no brightness or intensity). As a result, there's no slider control for this port, just an on/off button.

Once the LED is wired to the output, click the button to turn on the output.

The knocker output is one of those special "timed" outputs, with a timer circuit that cuts off power after a couple of seconds of continuous ON time. So you should see the LED light up for a couple of seconds and then turn off. If you click the button to turn the port off, and then click it again to turn it back on, the cycle should repeat.

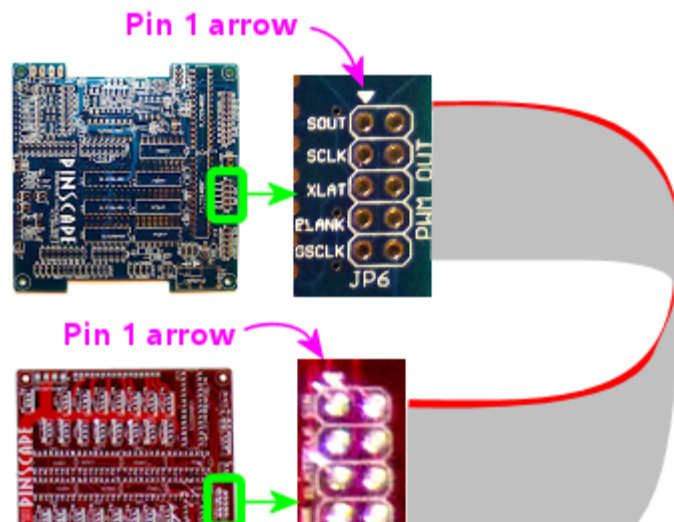
The timer *doesn't* set a *minimum* ON-time. It only sets a maximum. So if you quickly click the button ON and back OFF, the LED will turn off immediately; it doesn't stay on for the timer duration.

Test the power boards

If you've been following our procedures so far, you've now tested all of the major systems on the main board at this point, so you can move on to the power board. Before attaching the power board, power down the system and turn off the ATX power supplies.

Connect the power board to the main board via a 10-pin ribbon cable. See Chapter 83, Ribbon Cables if you haven't built this cable yet. This cable connects between **JP6 (PWM OUT)** on the main board and **JP2 (PWM IN)** on the power board.

Be sure to align pin 1 on both ends of the cable. I recommend marking a red stripe along one edge down the whole length of the cable, and calling that the Pin 1 side. That makes it easier to get the orientation right at both ends, since all you have to do is align the red stripe with the "Pin 1 arrow" marked on the board at each end.

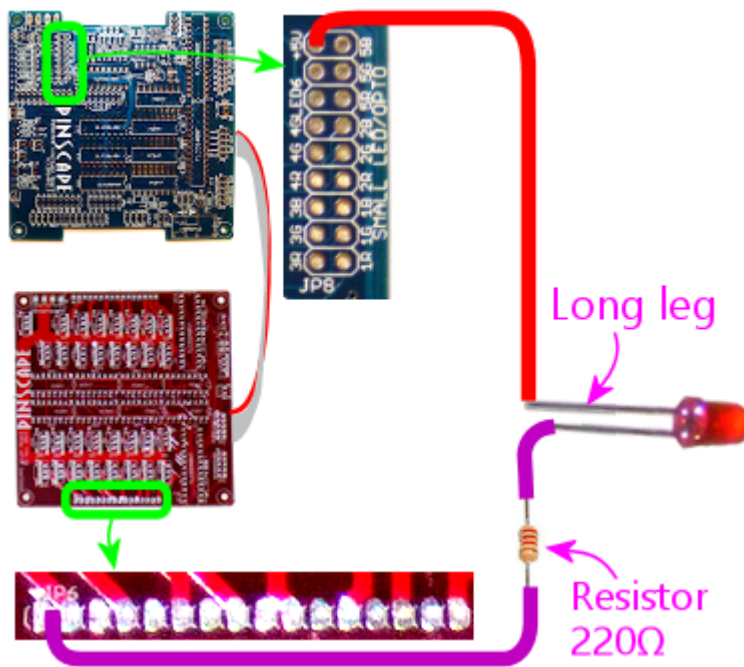


Also connect the two power connects (PC PSU and 2ND PSU) to the power board, exactly as on the main board. As always, **both** power connectors **must** be connected for the board to function properly.

Power up the PC and the ATX power supplies. As with the original main board test, pay close attention to any signs of overheating parts, and be ready to shut down the power supplies immediately if necessary.

Assuming that nothing is overheating, we can proceed with testing the power board outputs.

The test rig here is the same one we used for the flasher, strobe, and knocker tests on the main board. Keep one end of the LED attached to the +5V output on the RGB Flasher header on the main board.



You might wonder: why are we using +5V flasher pin from the *main* board to test the *power* board outputs? There's nothing magic going on here. The reason this works is that +5V flasher pin is simply a direct connection to the 5V rail on your secondary ATX power supply. You could just as well connect to the ATX 5V wire (the red wire in the disk power connectors) directly. That's more like what you're going to do with the actual deployed devices that you connect to the power board. But I think the 5V flasher pin is more convenient for this test. It's the same thing electrically, so you might as well use what's easiest.

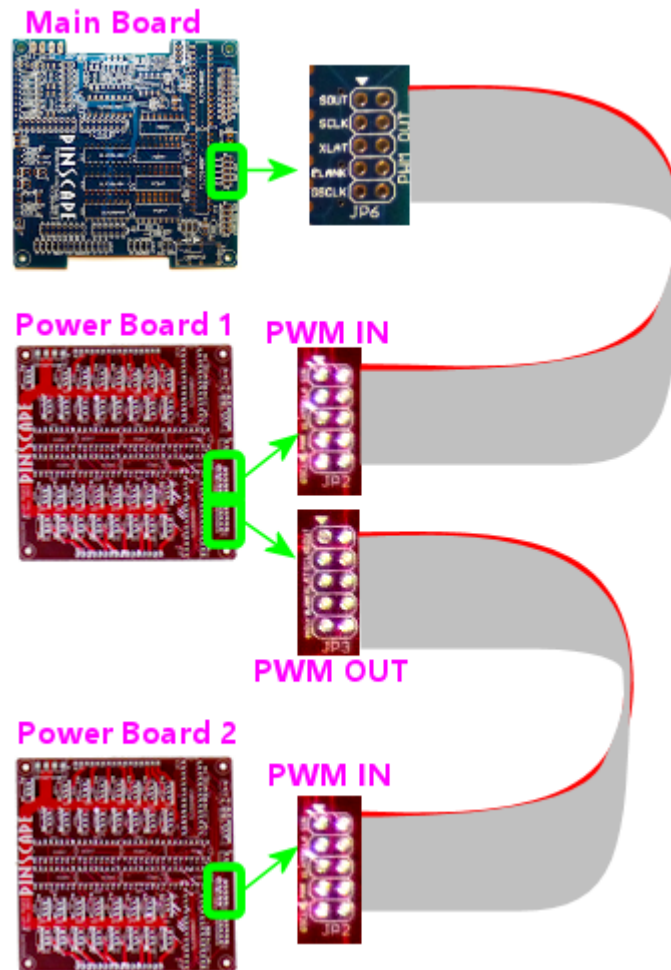
In the Config Tool Output Port tester, the power board ports are the ones with Pin labels **TLC5940 #3** and **TLC5940 #4**. #3 controls the first bank of outputs, on JP5. #4 controls the second bank of outputs, on JP6. The outputs are numbered across the pins, starting at the "Pin 1 arrow" on each header. In the standard setup, JP5 is assigned to Port No. 18 through 33, and JP6 is Port No. 34 through 49. But pay more attention to the TLC5940#3 and #4 in the Pin column - that's the way to identify these ports for sure.

As with the flasher ports, each one has a slider. When the LED is connected to a given port pin, sliding the port's slider up to 255 should turn the LED on at full

brightness.

Testing a second power board

If you have a second power board, the test procedure is very similar to the first. The only difference is that you're going to connect the "starting" end of the second power board's ribbon cable to the *first power board's JP3 (PWM OUT)* connector instead of connecting it to the main board. That forms the daisy chain from the main board to the first power board to the second power board.



As always, power everything down before connecting the data cables or power cables to the new board. And as before, monitor it for overheating parts for the first couple of minutes, just in case something's installed wrong.

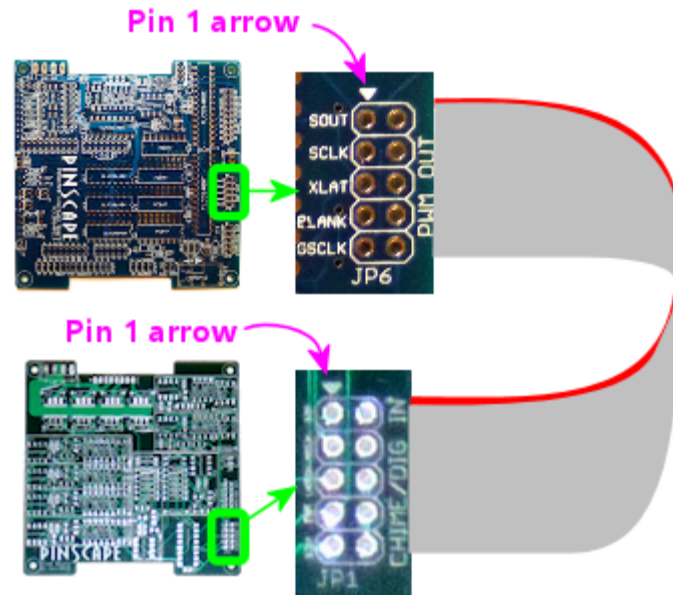
The second power board's ports will be the ones with Pin labels **TLC5940 #5** and **TLC5940 #6**. Otherwise, the test procedure is exactly the same as for the first power board.

Test the chime boards

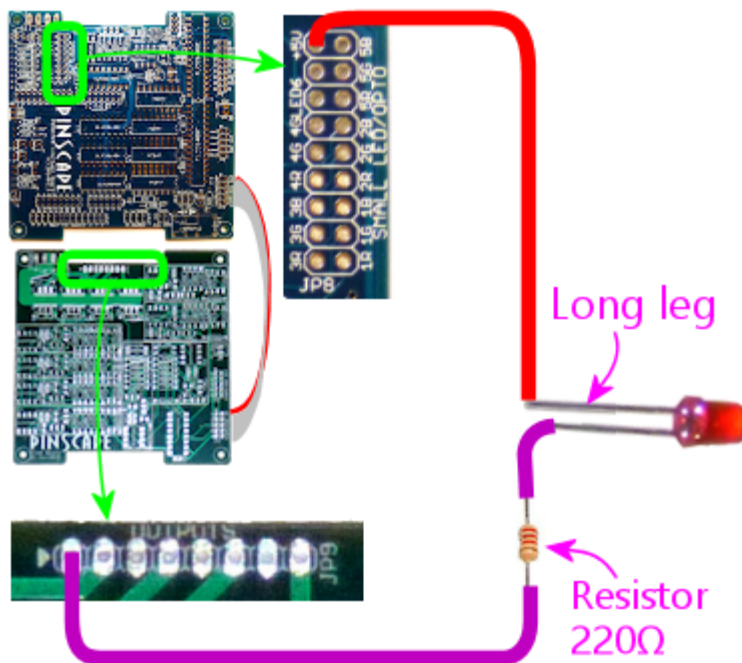
Testing the chime boards is very similar to testing the power boards. Power everything down before proceeding. Connect the PC PSU and 2ND PSU power cables - as always, both cables must be connected to each board.

Connect the ribbon cable between the main board's CHIME/DIG OUT port (JP5) and the chime's board's CHIME/DIG IN port (JP1). As with the power board ribbon cable, make sure that the cable is aligned so that Pin 1 on the main board header connects to Pin 1 on the chime board header. I recommend marking a red stripe on one edge

for the whole length of the ribbon cable to mark the Pin 1 side, to make it easier to get the orientation right when plugging it in.



We'll continue using the same test rig used for the power board ports and main board flasher ports. Keep the "+" end of the LED connected to the +5V flasher output on the main board, and move the "-" end over to the chime board outputs, one at a time.



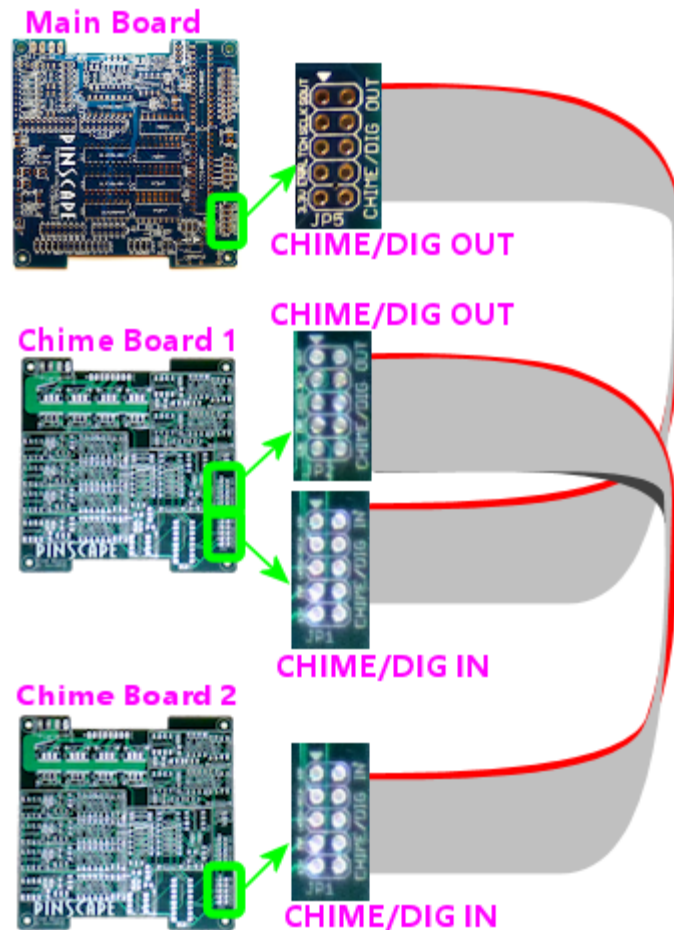
The chime board ports will be listed with Pin names starting with **74HC595 #1**. That's the digital controller chip that's used on these boards. There's one chip per board, so the first chime board attached will be 74HC595 #1. If you attach a second chime board, it'll be labeled 74HC595 #2.

The chime board outputs are "digital", meaning they're simple on/off outputs without PWM control. So the Output Port Tester just shows a simple on/off button for each port, just like the knocker port on the main board. And like the knocker port, these ports have hardware timers that will cut the power if the port is left on for more than a couple of seconds. As you test each port, you should see the LED light up for a

couple of seconds and then automatically shut off.

Testing a second chime board

The procedure for testing a second chime board is almost identical to testing the first one. The only difference connecting it is that you connect the starting end of its ribbon cable to the **CHIME/DIG OUT (JP2)** port on the *first chime board* instead of connecting it to the main board. And the only difference testing is that the output ports will have Pin names in the Output Port Tester starting with **74HC595 #2** for the second board (and #3 for the third board, and so on).



Test the IR transmitter and receiver

If you installed the IR devices, you can test them simply by running through the IR TV control setup procedure in Chapter 112, IR Remote Control. I don't have any more simplified tests to suggest, since there's nothing really to test other than their ability to receive and transmit signals, which the programming procedure will exercise.

97. Debugging the Expansion Boards

This section has some things you can try to diagnose and fix any problems you've having with a set of expansion boards.

If you haven't already, you should run through the test procedures in Chapter 96, Initial Expansion Board Testing. That should give you a clearer idea of what's working and what's not working. That section also includes some basic debugging steps that might solve some simpler problems.

If you go through all of the debugging suggestions for a particular symptom without identifying the problem, let me know. I'd like to include as many causes and fixes as possible, so if you come up with a new one that I haven't covered here, I'd like to add it.

Top problem sources

Here are the main causes of problems I've seen when helping people debug expansion boards:

- Bad solder joints. These boards have lots and lots of solder pads, so it's very easy for a couple of them to fail. A so-called "cold solder joint" is one that looks good visually, but has tiny fractures in the solder, blocking electrical contact. The best way to test for these is continuity testing (see below); we'll point out which points in the circuits you should test for each failure mode.
- Defective chips. Some chips are simply defective when they roll off the factory assembly line, and some get damaged during shipping or storage. I've had defective chips a couple of times even when ordering from Mouser, but it seems to happen a lot more with eBay. With Mouser, you can count on them to sell you authentic, factory-direct parts, and to handle them properly at their warehouse, but a few dead parts still slip through the cracks. With eBay, who knows what you're getting. I've heard from several people who had problems with their boards that turned out to be dead TLC5940 chips that they ordered from eBay. You can't get those chips anywhere but eBay these days, so you just have to hope for the best. I've also run into a couple of bad optocouplers (PC817 and PC847). Unfortunately, there's no easy test procedure you can perform for the TLC5940 or PC8x7 chips individually (and you shouldn't have to do that anyway, since you should be able to count on the factory to do quality control checks). But there are some things you can do once they're installed to trace a problem to a particular dead chip, which we'll get to shortly.
- Simple chip insertion or cabling errors. See "quick sanity checks" below.

Quick sanity checks

At the risk of belaboring the obvious, it's always good to double-check some of the basics:

- Make sure all power cables are connected. The PC PSU and 2ND PSU connectors must **both** be connected for the boards to function properly.
- Make sure cables are inserted the right way. For ribbon cables, I always recommend marking one edge of the cable with a red stripe (or something similar), and making sure that the red stripe is on the side with the "Pin 1" marking on the circuit board connector at both ends.
- Make sure IC chips are inserted the right way. See Chapter 79, IC Chips for

how to properly orient each type of IC used in the project.

- If you're using sockets for the IC chips, visually inspect them and make sure the chips are fully seated, and that all of the pins are inserted. Make sure none of the pins are sticking out the side of the socket or got folded under the chip.

Visual solder check

I recommend giving the board a careful visual inspection with a magnifying glass, looking for suspicious solder joints. This is an easy test, and it's pretty effective - I can usually catch one or two bad connections this way after first assembling a board. Some bad solder joints can evade visual detection, so this probably won't catch them all, but it can save you some time later if you catch the obvious ones up front.

Continuity testing

One of the main tests we'll recommend throughout this section is "continuity testing," which means that you use a multimeter to check that there's a solid connection between two points.

I think the most common cause of problems I've seen with the expansion boards is bad solder joints. It's no wonder, since these boards have quite a lot of individual pin pads, and I think even the most experienced soldering experts still have some percentage of "cold" (non-connected) solder joints on the first attempt. Every time I've built these boards myself, I've had two or three bad connections per board.

The tricky thing about cold solder joints is that they can look perfectly normal to visual inspection. The solder can sometimes form tiny fractures as it cools that aren't readily visible but are big enough to prevent electrical contact. So the best way to find them is to test for electrical continuity. We'll point out specific places to test for each set of symptoms.

Most multimeters have a continuity testing mode that makes this fairly easy. The way this usually works is that the meter beeps when it detects good continuity, so you can test connections between several points quickly without having to stop and look at the meter. A slower alternative if you don't have a proper continuity-test mode is to use the resistance mode (to measure Ohms). A good connection should show a very small resistance value, well below 1Ω , and usually something like $.01\Omega$.

Always test continuity with the power OFF.

When testing continuity to an IC pin, always test (if possible) at the **pin** itself. Not the solder pad on the bottom of the board, but rather the metal leg coming out of the chip, on the top side of the board. That's the surest place to test. If the problem is a cold solder joint, the whole problem is that the solder hasn't adhered properly to the pin. It might have adhered properly to the copper trace on the circuit board, so you might see continuity to the little glob of solder. But that doesn't matter if the solder isn't attached to the pin. So if you measure at the solder, you might see good continuity, even if the pin isn't attached properly. So measure at the pin when possible. The same goes for resistors, capacitors, transistors, and so on. Likewise, when testing a pin header, measure directly at the pin on the top side.

If a continuity test fails, the best and easiest remedy is to get out your soldering iron and re-melt the solder at each end of the connection. You might also apply fresh solder, but I'd only do that if the current joint looks starved for solder; you don't want a giant glob that might create a short circuit with an adjacent pad.

After re-soldering to try to fix a failed test, check continuity again. If it looks good,

go back and repeat the **live** test that led you to do the continuity test in the first place - test the button input, output port, etc. Most problems are due to single continuity errors, so re-soldering the pins when you find such a problem will usually fix the whole problem.

EAGLE plans

For trickier problems, we might ask you to open the plans for the boards in the EAGLE software, which is the circuit board design program used to create them. I know - that might sound scary. But the tasks needed for this section are pretty simple, and we'll walk you through them. The EAGLE files are the definitive source of information on the details of how the parts on the board are connected, so that's really the most reliable way to get some of the needed information when tracing connections.

If you do get to a point where we ask you to go to the board plans, here's how to set that up:

- Download and install the free personal-use version of EAGLE from Autodesk:

www.autodesk.com/products/eagle/overview

- Download and UNZIP the board plans:

mjrnet.org/pinscape/expansion_board/download.php

- Launch EAGLE
- In the EAGLE control panel menu, select **File > Open > Board**
- Select the .BRD file for the expansion board you want to look at

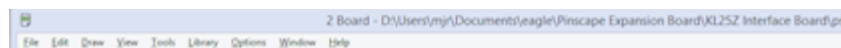
A few EAGLE usage tips

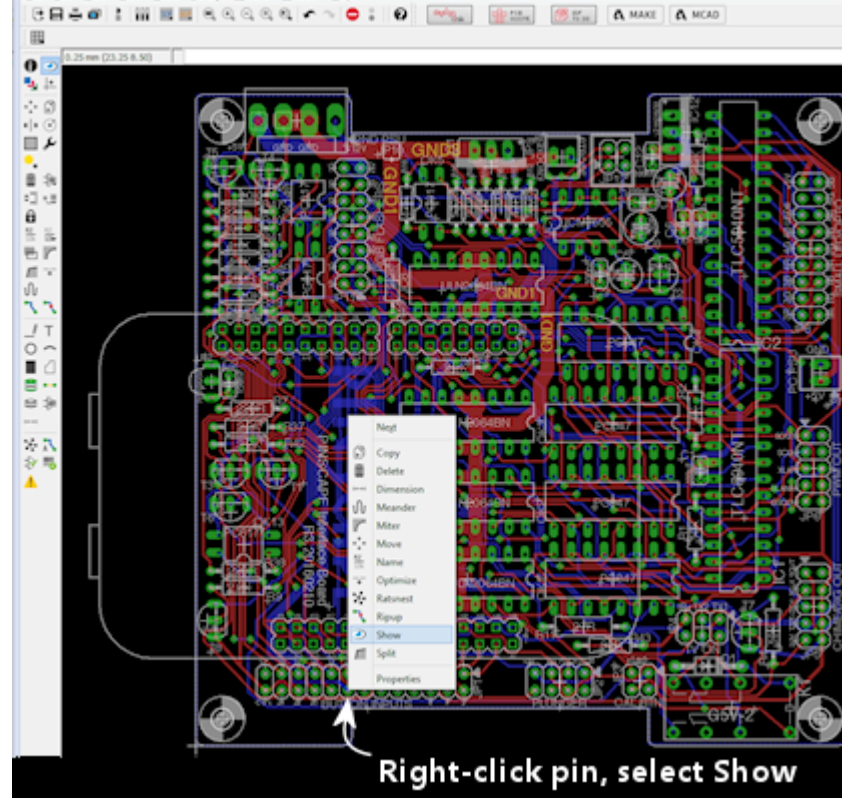
- The .BRD (board) file represents the layout of the board, as a 2D top-down view; everything is in the same place as on the physical boards
- Spin the mouse wheel to scroll in and out
- Press and hold the mouse wheel to pan the circuit board view
- Right-click on pins or circuit traces for a context menu with possible commands for the object
- On the main menu, select **File > Switch to schematic** to view the circuit schematic
- For help reading the schematic, see Chapter 70, Schematics

How to trace a circuit in EAGLE

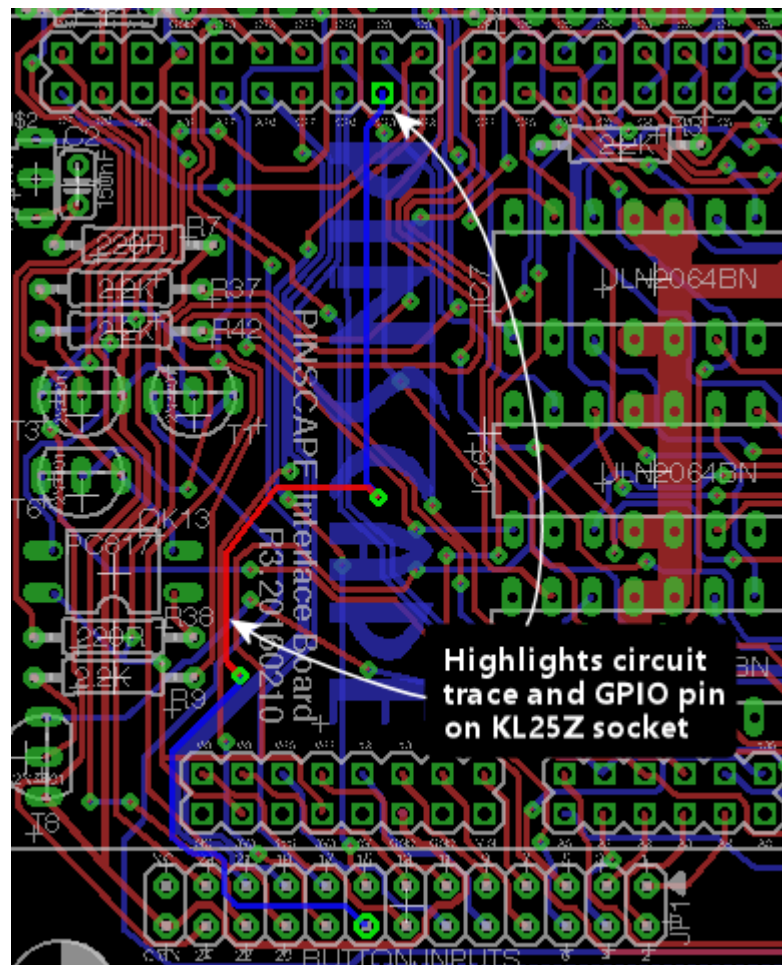
EAGLE makes it easy to see exactly what's connected to a particular point on the circuit board. This is the best way to trace a connection.

- Run EAGLE and load the .BRD file you want to look at as described above.
- Find the header pin, IC pin, or other component connection point you want trace
- Right-click on the center of the pin
- Select **Show** from the context menu





- The network of connections to that pin will light up, including the copper traces on the circuit board, and the IC/header/component pins connected at the other end or along the way. For example, here's the connection from a pin on the Button Input header to the KL25Z sockets:



Button inputs

Test button switch with the Button Tester window in the Config Tool - not, say, by checking for Windows keyboard or joystick input. The Button Tester window is a more direct view of the hardware input.

Double-check the System Type setting in the Config Tool settings to make sure it's set to Pinscape Expansion Boards.

No buttons are working

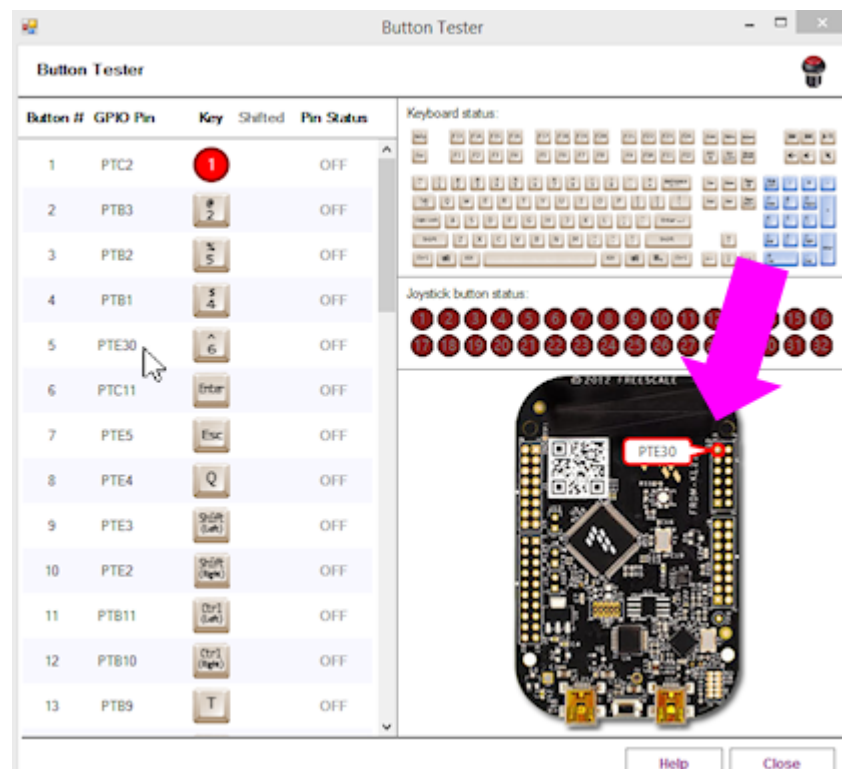
If none of the button inputs are working, the CMN (Common) pin on the button input header is the likely culprit, because that's the one connection that all of the inputs share. Check continuity from the CMN pin on the header to a ground pins on the KL25Z, such as J9 pins 12 or 14 (see).

Some buttons are working, some aren't

If button #6 is the only one that's not working, there's a very good chance that your System Type in the settings is set to Standalone KL25Z. If so, change it to Pinscape Expansion Boards.

The individual button input pins connect directly to KL25Z GPIO pins, so really the only thing that can go wrong other than software configuration problems is cold solder joints. Check continuity on each non-working button as follows:

- Use the Button Tester window to identify the GPIO port for the non-working pin. The number in the **Button#** column corresponds to the pin number on the expansion board header. The **GPIO Pin** column tells you the KL25Z GPIO port. If you move the mouse over the button's row, the location of the GPIO pin will be highlighted on the KL25Z diagram on the right.



- With the power off, test continuity from the pin on the Button Inputs pin header (testing the pin on the top side of the board) to the KL25Z pin highlighted in the diagram.

- If the continuity is bad, try re-soldering the Button Input header pin and the corresponding KL25Z socket pin on the expansion board. While you're at it, re-solder the header pin connection on the KL25Z itself - it might be bad at the KL25Z end.
- If the continuity is good, you might try re-soldering the KL25Z header pin connection anyway, as you can't really check continuity all the way to the KL25Z itself - only to the pin. So the problem could still be a cold solder join in the KL25Z pin header.
- If none of that helps, go back and check the software configuration again, and cross-check it against the EAGLE plans for the board:
 - Check the GPIO pin assignment in the Config Tool
 - Open the .BRD file for the main boards in EAGLE
 - Find the pin of interest on the Button Inputs header
 - Right click in the center of the pin and select **Show** from the menu
 - That'll light up the whole network of things connected to that pin, including the traces on the circuit board and the KL25Z socket pin on the other end. Zero in on the KL25Z socket pin. Make sure it's in the same position as the one you've been testing.
 - See "How to trace a circuit in EAGLE" above for more on this.

Feedback device outputs

Test feedback outputs using the Output Port Tester in the Config Tool, not DOF or LedBlinky or other third-party programs. The Output Port Tester is the most direct test of the hardware, bypassing any configuration problems with other software. DOF in particular has lots of its own things that can go wrong; you don't want to have to guess whether the problem is in the software or the hardware.

Main board flasher/small LED ports: none are working

The RGB Flasher ports and Strobe port are controlled by the first TLC5940 chip, labeled IC1 on the circuit board. The Small RGB LED ports are controlled by the second TLC5940 chip, IC2.

If all 32 of these ports are dead, the most likely cause is a bad connection to one or more of its main data or power inputs. These chips have several critical inputs that all have to be working for the chips to function, so a single continuity problem in any of the input pins will make the entire chip appear to be dead.

Another possibility is that one or both TLC5940 chips are defective - the chips are daisy-chained, so a defective IC1 can make both chips appear to be dead.

If you installed the TLC5940 chips in sockets, and you have additional TLC5940 chips on hand (spares, or the chips installed in a power board), it's easy to swap the installed pair for another pair to check for bad chips. Even though the DOA rate for these chips is fairly high with eBay sources, I don't think I've talked to anyone who had more than one or two bad chips in a batch. So if you bought at least four or five of them, the odds are good that *some* of them will work. So:

- First, try swapping IC1 and IC2 with one another; do another test run
- If everything's still dead, try replacing both IC1 and IC2 with spares

If that didn't help, it's still possible that you got an entirely bad lot of TLC5940 chips, but hopefully not - I haven't heard from anyone who's had that happen yet. It's more

likely that at least some of your chips are good, and that the problem is instead in the wiring.

Make the following continuity tests. I'd recommend having the .BRD file open in EAGLE while you're doing this, since that makes it a lot faster to find the test points on the physical boards. If you prefer, you can refer to the Appendix 1, KL25Z pin-out diagram to locate the pins on the KL25Z. When we say something like "J1 pin 11" on the KL25Z, we're talking about the location KL25Z's pin headers, not the expansion board headers.

- Pin 17 on **IC1 only** should connect to PIN 26 on IC2
- Pin 18 on IC1 and IC2 should connect to KL25Z PTA1 (J1 pin 2 on the KL25Z)
- Pin 19 on IC1 and IC2 should connect to the GND pin on the PC PSU connector
- Pin 22 on IC1 and IC2 should connect to the GND pin on the PC PSU connector
- Pin 23 on IC1 and IC2 should connect to KL25Z PTC7 (J1 pin 1 on the KL25Z)
- Pin 24 on IC1 and IC2 should connect to KL25Z PTC10 (J1 pin 13 on the KL25Z)
- Pin 25 on IC1 and IC2 should connect to KL25Z PTC5 (J1 pin 9 on the KL25Z)
- Pin 26 on IC1 should connect to KL25Z PTC6 (J1 pin 11 on the KL25Z)
- Pin 26 on **IC2 only** should connect to Pin 17 on IC1 (as already noted above)
- Pin 27 on IC1 and IC2 should connect to the GND pin on the PC PSU connector

If all of that looks good, check that the chip's power supply is working. With the power on, carefully measure DC voltage at pin 21 (on both chips). Connect the meter's red probe to pin 21 on the chip, and connect the black probe to one of the PWM OUT pins in the row nearest the edge of the board. This should read 3.3V. If not, check the pin 21 solder connection, and check all of the solder connections for IC12 (the LD1117AV33 chip that looks like a MOSFET).

Main board flasher/small LED ports: no flasher ports work, some/all Small LED ports work

The most likely problem here is that the +5V connection on the flasher header isn't connected properly to power. Check continuity between that pin and the +5V pin on the 2ND PSU header (JP10). If that looks good, try measuring the voltage at the flasher +5V pin, carefully, with the power on. Connect the red probe of your meter to the flasher +5V pin, and connect the black probe to one of the middle pins (pin 4 or 5) on IC8. If that doesn't read 5V, the problem must be in your 2ND PSU power cable - check its connections carefully.

The next possibility, although it's really unlikely, is that IC1 is defective. Given that IC2 must be working for the Small LED ports to work, we can easily test this possibility by swapping IC1 and IC2. If the working and non-working ports trade places, the problem is a defective chip - the one in IC2 after the swap. If nothing changes, both chips are probably good, and the problem is in the wiring.

Go back to the "none are working" section and check the power inputs to IC1. You can also check the signal inputs, but it's unlikely that IC2 would be working at all if any of IC1's signal inputs aren't working, since IC2 takes its signal input from IC1's output, so a broken IC1 will usually prevent IC2 from working either.

Main board flasher/small LED ports: some/all flasher ports work, no Small LED ports work

If some of the ports are working, but **all** of the Small LED ports are broken, the

problem is probably the second TLC5940 chip, IC2. This is a direct corollary to "none are working" above, but isolated to IC2. The main things to look at in this case are continuity to the IC2 data and power inputs, and the possibility of a defective IC2.

One other, simpler possibility is that the +5V pin on the Small LED header isn't working. Check continuity between that pin and the +5V pin on the PC PSU header (JP7). You might also want to check the DC voltage reading on that pin with the power on - to read that voltage, connect the red meter probe to the +5V pin on the Small LED header, and connect the black probe to one of the Ground pins on JP6 (PWM OUT) - any of the pins on that header in the outer row (closer to the edge of the board). If continuity to the PC PSU header is good, the problem must be a bad connection in your power cable.

If you installed the chips in sockets, we can easily check to see if IC2 is bad by swapping IC1 and IC2. We know that IC1 is a working chip, since some or all of its ports are working. If we swap IC1 and IC2, then, we move the known-working chip into the IC2 socket, and the questionable chip into the IC1 socket. So power down, swap the chips, and run the port tests again. Here's what we'll learn:

- If all of the ports are dead now, or the live and dead ports trade places (the working ports move to the Small LED group, and all of the flasher/strobe ports are now dead), you have a bad chip - the one that's now in the IC1 socket is the dead one, and the other one is good. Throw out the chip in the IC1 socket and try a new one.
- If the same set of ports remain working as before, the chips are both good, so the problem is in the wiring. Proceed with the continuity checks below.

If the problem is in the wiring, it's most likely either the power connections to IC2, or the data connections to IC2. Follow the same procedures to test continuity to the TLC5940 inputs described above under "none are working", but you should only have to look at the IC2 inputs in this case.

Main board flasher/small LED ports: some work, some don't

If some of the flasher and small LED ports are working and some aren't, it's almost certain that the TLC5940 chips are working properly and that all of their power and data inputs are working properly. The chips and their inputs have to be working for *any* of the ports to work, so we can rule out those sorts of problems if even a single port works correctly.

If you installed the TLC5940 chips in sockets, one easy test that you can do to rule out a partially defective chip (mostly working, but with a few dead ports) is to swap IC1 and IC2 with one another. If the bad ports followed the chips, you must have a chip with one or more bad ports. I haven't talked to anyone who's encountered such a thing, but it seems like a possibility in principle. If the dead ports are exactly the same before and after the swap, the chips must be good - it's the wiring.

Checking the small LED output wiring

The Small LED outputs are connected directly to the TLC5940 output ports, so if one of those ports isn't working, it's probably just a bad solder joint at either the Small LED header pin or the corresponding TLC5940 pin. To identify the path to trace:

- Launch EAGLE and load the .BRD file
- Right-click in the center of the Small LED header pin of interest
- Select **Show** from the context menu
- This will highlight the circuit trace from the Small LED header pin to the

TLC5940 pin for the port. The highlighted TLC5940 pin is the one to check. Test continuity between that pin and the header pin.

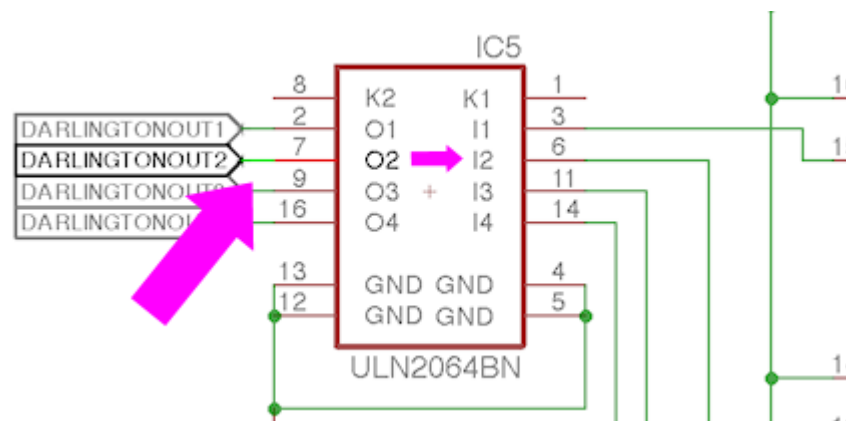
- If there's no continuity, re-solder the connection at each pin (the header pin and the TLC5940 pin)
- If you're using sockets, also check to make sure that particular TLC5940 pin is properly inserted in its socket

Checking the flasher/strobe output wiring

The flasher/strobe ports are more complex to trace than the Small LED ports, because the former are connected through optocouplers and Darlington transistor chips to allow them to drive larger devices. The problem for a flasher port could therefore be a continuity problem along the chain of connections through those other components, or it could be that one of those components is defective.

You'll need the EAGLE .BRD plans to trace these connections:

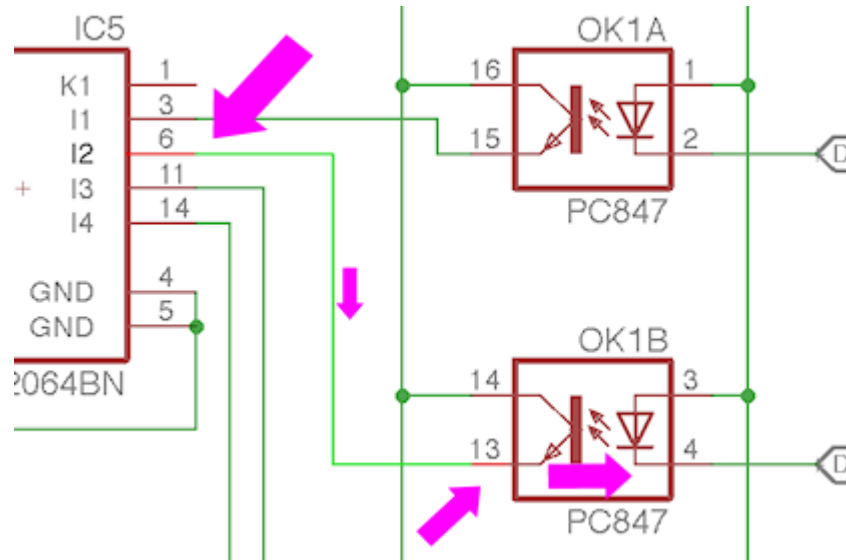
- Launch EAGLE and open the .BRD file
- Right click on the RGB FLASHERS header (JP11) pin or STROBE (JP9) pin that you want to trace
- Select **Show** from the context menu
- This will highlight a trace leading back to one of the ULN2064BN chip pins. Continuity check the connection between the header pin and the ULN2064BN pin. If that's bad, re-solder both pins - this might be the whole problem, so do a new power-on test of the output.
- For the next step, you'll need the schematic window to be open: on the main menu, select **File > Switch to schematic**
- In the "Sheets" list on the left, select Sheet 3 ("Flasher/Lamp Outputs")
- Click on the Board window's title bar to bring it to the front, and **Show** the pin again
- Click on the Schematic window's title bar to bring it to the front. The same pin should be highlighted on the schematic in two places: on the JP11 header and on one of the ULN2064BN chips. Find the highlighted ULN2064BN pin - this should be labeled "O1", "O2", "O3", or "O4" inside the ULN2064BN box.
- Trace across the ULN2064BN box to the corresponding "I" pin on the opposite side - from "O1" to "I1", "O2" to "I2", etc.



- Right-click the red line sticking out of the ULN2064BN box for that pin and select **Show**
- Click on the Board window's title bar to bring it to the front. You should now

see two new pins highlighted: a ULN2064BN pin, and a pin on a PC847 chip. Check continuity between those two pins. If it's bad, re-solder both pins - this might fix it, so do a new power-on test of the output.

- Still not there? Click on the Schematic window's title bar to bring it to the front, and find the PC847 pin that's now highlighted as part of the circuit trace. This should be the "emitter" of a PC847 - a line with a little arrow pointing out of the box. Find the pin on the opposite side of the PC847 box - right click it and select **Show**.



- Go back to the Board view. Yet another pair of pins should be highlighted: one on the PC847, and one on the TLC5940. Check continuity between these pins. If it's bad, re-solder both pins, and do another power-on test of the output.

We've now traced the circuit all the way from the pin header to the TLC5940, so if it's a continuity problem, you should have found it by now. The remaining possibility is that one of those two components we just traced is bad - either the ULN2064BN or the PC847. (Hopefully you paid attention in the steps above to which ULN2064BN and which PC847 we were tracing through. If not, go back through the tracing steps to identify the chips. Write down their chip numbers so that you know which ones to look at more closely.)

If you installed these chips in sockets, the easiest test is always to do a chip-swap. Do one swap at a time: first swap the ULN2064BN with one of the others on the board, or with a spare if you have one. Run the live test again. If that doesn't fix it, swap the PC847 with one of the others on the board or with a spare.

Main board knocker port isn't working

The knocker port is a pretty complex one to trace because of the timer circuit. For whatever reason, though, this one doesn't seem to give anyone any trouble. The parts in this circuit are simple ones that are likely to be reliable, so any problems are probably due to a bad solder connection. Rather than trying to trace through the many connections one by one, I'm going to just point you to the parts to check. Check the solder joints on each part and re-solder any that look suspicious. On the circuit board, these parts are mostly clustered around the knocker/strobe output header, but a few are scattered elsewhere.

- C5
- C7

- C8
- C9
- IC11
- OK5
- R6
- R8
- R10
- R12
- R13
- R14
- R18
- R37
- T2
- T3
- Q1

For the definitive list of parts, look at the EAGLE schematic, on Sheet 2 ("Knocker Output").

Power board ports: none work

The power board ports are controlled by TLC5940 chips, like the flasher and small LED ports on the main board, so the diagnosis process is similar. If all of the ports on this board are dead, the main possibilities are:

- The data cable from the main board isn't connected properly. Visually check that it's installed properly and that both ends are oriented properly. Make sure that the same edge of the cable aligned with the "Pin 1" arrow on the main board is aligned with the "Pin 1" arrow on the power board. Make sure that it's plugged in the PWM IN port on the power board, and the PWM OUT port on the main board. (The PWM IN and PWM OUT ports on the power board look identical, so read the label and make sure you have the IN port on the power board. Likewise, the PWM OUT and CHIME OUT ports on the main board look identical. Make sure you're using the PWM OUT port on the main board and not the CHIME OUT port.)
- If the data cable looks to be installed properly, check continuity (with the cable connected, and power off) for each pin on the headers. Unlike the usual rule about testing the pin, this time you want to check continuity on **bottom** of the board, testing at the solder pads. The reason is that we're testing the cable this time, so we want to make sure we have a good connection from solder joint to solder joint. Check pin #1 on the main board PWM OUT against pin #1 on the power board PWM IN, pin #2 against pin #2, etc. All pins must be connected.
- Assuming you have all of the outputs on the main board working at this point, and assuming you installed the TLC5940 chips on both boards in sockets, you can test for a defective chip by swapping the TLC5940 chips on the power board with the ones on the main board. We know that the ones on the main board must be working if their outputs are working.
- If the chip swap doesn't change anything, test continuity to the data connections. This is similar to testing the TLC5940 pins on the main board:

- Pin 17 on **IC1 only** should connect to PIN 26 on IC2
 - Pin 18 on IC1 and IC2 should connect to KL25Z PTA1 (J1 pin 2 on the KL25Z)
 - Pin 19 on IC1 and IC2 should connect to the GND pin on the PC PSU connector
 - Pin 22 on IC1 and IC2 should connect to the GND pin on the PC PSU connector
 - Pin 23 on IC1 and IC2 should connect to KL25Z PTC7 (J1 pin 1 on the KL25Z)
 - Pin 24 on IC1 and IC2 should connect to KL25Z PTC10 (J1 pin 13 on the KL25Z)
 - Pin 25 on IC1 and IC2 should connect to KL25Z PTC5 (J1 pin 9 on the KL25Z)
 - Pin 26 on **power board IC1 only** should connect to Pin 17 on **main board IC2** (note that we're testing ICs on separate boards here!)
 - Pin 26 on **IC2 only** should connect to pin 17 on IC1
 - Pin 27 on power board IC1 and IC2 should connect to the GND pin on the PC PSU connector
- If all of that looks good, check that the chip's power supply is working. With the power on, carefully measure DC voltage at pin 21 (on both chips). Connect the meter's red probe to pin 21 on the chip, and connect the black probe to one of the PWM OUT pins in the row nearest the edge of the board. This should read 3.3V. If not, check the pin 21 solder connection, and check all of the solder connections for IC12 (the LD1117AV33 chip that looks like a MOSFET).

Power board ports: some work, some don't

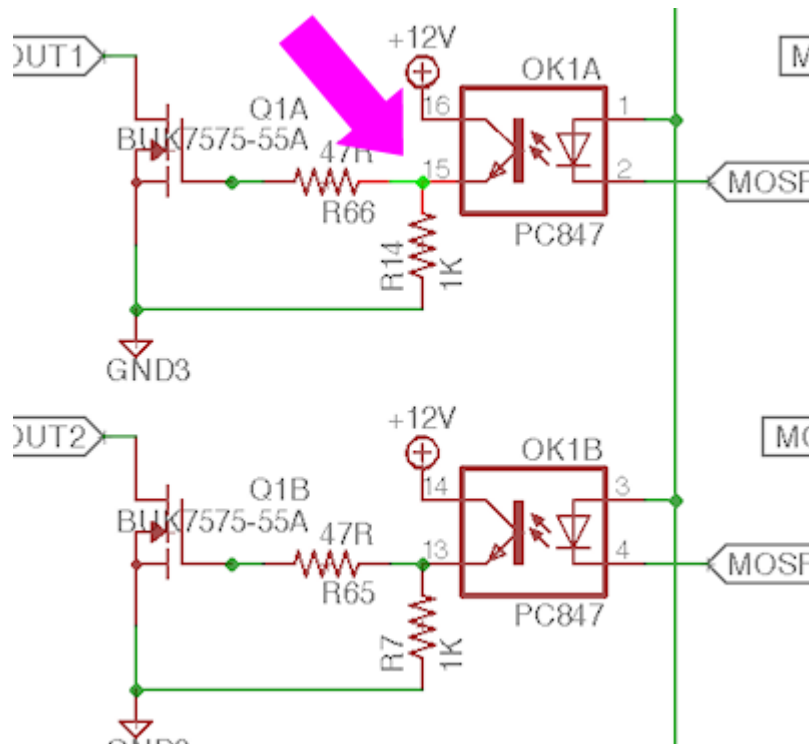
As with the main boards, if some ports are working, the data and power inputs to the TLC5940 chips must be good, since these chips won't work at all if there's a problem in any of those inputs. So the problem is either an isolated port failure in one of the TLC5940 chips (unlikely - I haven't heard of a case of this actually happening in the wild, only complete failures of the entire chip), or more likely, a problem in the wiring between the TLC5940 and the output port pin.

If you installed the TLC5940 chips in sockets, you can do a quick test to rule out the bad chip scenario by swapping chips. Simply swap the two TLC5940 chips with one another, and do another power-on test. If the pattern of bad ports remains the same, the chips are okay; if the pattern changes after swapping the chips (in particular, if the bad port follows the chip to a new port), the chip controlling the bad port probably has a bad output. Try replacing it with a fresh TLC5940 chip.

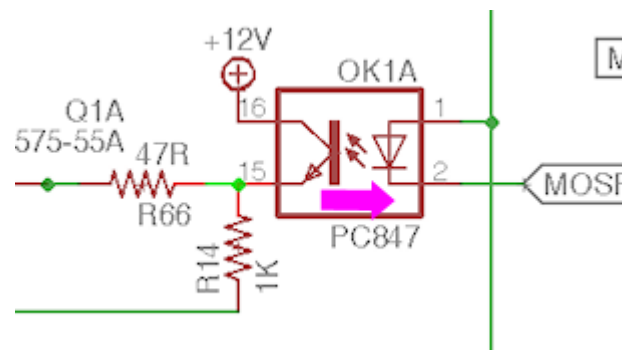
If the chip swap doesn't turn up anything, we have to trace the output circuit from the TLC5940 pin to the output pin header. You'll need to open the .BRD file in EAGLE for this process. As always, if you find a continuity problem, re-solder the pads at each end of the connection, re-check continuity, and if it looks good, run another live test to see if the problem is resolved.

- In the EAGLE Board view, right-click the pin on the HI POWER OUTS header that you want to trace, and select **Show** from the context menu
- This will highlight a trace back to one of the MOSFETs. Test continuity between the two points (the pin and the MOSFET - try to test at the MOSFET leg on the top side of the board if possible).

- Next, test the other two MOSFET legs. One will be connected to the super-wide ground trace; test continuity from that MOSFET leg to the GND pins on the 2ND PSU power connector (JP4). The other will connect to a resistor; check that connection from the MOSFET leg to the resistor leg.
- The other end of that resistor will connect to **two points**: another resistor right next to it, and a pin on a PC847 chip. Check both connections.
- At this point we have to consult the schematic to find the next jump. On the main menu, select **File > Switch to schematic**. In the "Sheets" list on the left, select page 2 ("MOSFET Outputs").
- Click in the Board window's title bar to bring it back to the front. **Show** the connection to the PC847 again. Note the chip number (OK1, OK2, etc). Click on the Schematic window's title bar to bring *it* to the front. Find the same OK*n* chip, and find the highlighted pin. It will be hard to spot because it's just a short wire segment, but it'll connect to the "emitter" side of one of the OK's.



- Trace across the OK chip from the highlighted emitter pin side to the pin on the opposite side of the box. Right-click that pin and select **Show**.

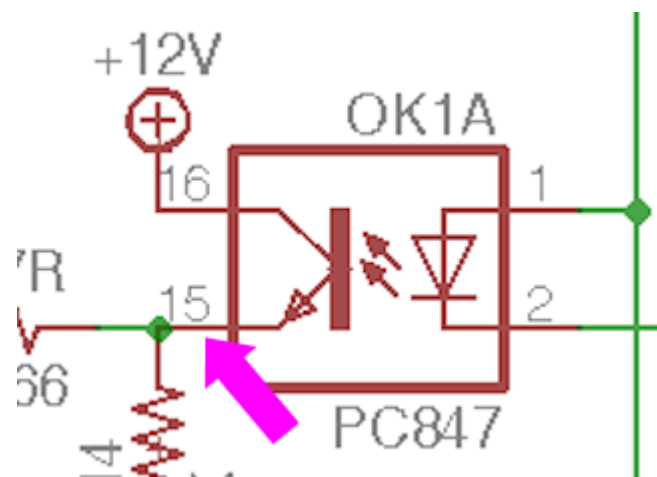


- Click on the Board window title bar to bring it back to the front. You should see a highlighted trace from a highlighted PC847 pin to a highlighted TLC5940 pin. Check continuity between those two pins.

That's the whole circuit, so if it's a continuity problem, it should have turned up somewhere in that process. If there are no continuity problems, you must have either a bad PC847 or a bad MOSFET. If you installed the PC847 chips in sockets, try swapping the suspect PC847 with one of the others, and do another live test.

Replacing the MOSFETs is more difficult since there's no good way to socket them, but at least they only have three pins, so it's possible to de-solder them with some patience. If you want to do a more definitive live test of the MOSFET before going to the trouble of de-soldering it, it's possible, but a little tricky. You have to do this as a live test with the power on, so be really careful about touching wires anywhere other than the exact points we say to:

- Go back to the schematic and find the PC847 controlling the MOSFET, following the procedure above to trace through the circuit.
- Identify the pin shown below on the schematic. Use the right-click **Show** command on each pin to highlight it and identify the physical pin location on the Board view. This is the first pin we're going to short in our test.



- In the board view, type into the command box at the top **show +12V** and press Enter. That will highlight a bunch of pins on each PC847 - one of them should be right next door to the one you identified above. This is the second pin we're going to short.
- With the power running, carefully short the two pins identified above. If the MOSFET is working, the output should turn on. If it doesn't, the MOSFET is probably dead, so I'd try replacing it.

Chime board ports: none work

There are two main "single points of failure" that would make the whole chime board fail to work:

- The data cable
- The 74HC595 chip

Start with the data cable, since it's easy to check. Make sure that it's installed with the same edge facing the Pin 1 arrow on the main board and the chime board. (I recommend the "red stripe" trick to make that easier: mark a red stripe down the whole length along one edge, and check that the red stripe aligns with the Pin 1 arrows marked on both boards.)

Make certain that the cable is plugged into the **CHIME/DIG OUT** port on the main board and the CHIME/DIG **IN** port on the chime board. Check the markings carefully

- the OUT port on the chime board looks exactly like the IN port. And the CHIME port on the main board looks exactly like the PWM port; make sure you're using the CHIME port with the chime board.

If you're sure the cable is installed correctly, check continuity across the pin headers connecting to the cable. Do this with the cable connected and the power off. Unlike the usual rule about testing at the pin, this time you want to check continuity on **bottom** of the board, testing at the solder pads. The reason is that we're testing the cable this time, so we want to make sure we a good connection from solder joint to solder joint. Check pin #1 on the main board CHIME/DIG OUT against pin #1 on the chime board CHIME/DIG IN, pin #2 against pin #2, etc. All pins must be connected.

The next thing to check is the 74HC595 chip. This controls all of the outputs, so if none of them are working, the problem could be a defective chip or a bad data or power connection. The chip won't work at all if the data or power connections aren't working.

If you installed the chip in a socket, and you have a spare, you can swap the chip with the spare, as an easy way to check for a defective chip.

To test the data connections to the 74HC595, leave the data cable to the main board attached, so that we can test continuity all the way back to the KL25Z pins:

- Pin 8 on the chime board 74HC595 should connect to the GND pin on the PC PSU connector (JP7)
- Pins 10 and 16 on the chime board 74HC595 should connect to the 3V3 pins on the KL25Z (J9 pins 4 and 8)
- Pin 11 on the chime board 74HC595 should connect to PTA4 on the KL25Z (J1 pin 10)
- Pin 12 on the chime board 74HC595 should connect to PTA12 on the KL25Z (J1 pin 8)
- Pin 14 on the chime board 74HC595 should connect to PTA5 on the KL25Z (J1 pin 12)

The next test will require voltage testing with the power on, so be careful - don't let your multimeter probes short any pins together while working.

- Power up the system for a live test
- Connect the multimeter's black probe to one of the Ground pins on the CHIME/DIG OUT connector - the pins on the side nearest the edge, **except for pin 2** (the one near the pin 1 arrow), which isn't connected
- Use the red probe to measure the voltage
- Pin 13 on the 74HC595 should read a low voltage, about 0.6V

If pin 13 reads zero voltage, or nearly zero, or weird random fluctuating voltages, there might be a continuity problem - check the connection to the pin 7 (ENA) on the CHIME/DIG IN connector on the chime board. If it reads about 3.3V, the problem is probably on the main board side. Check wiring to the following **main board** parts:

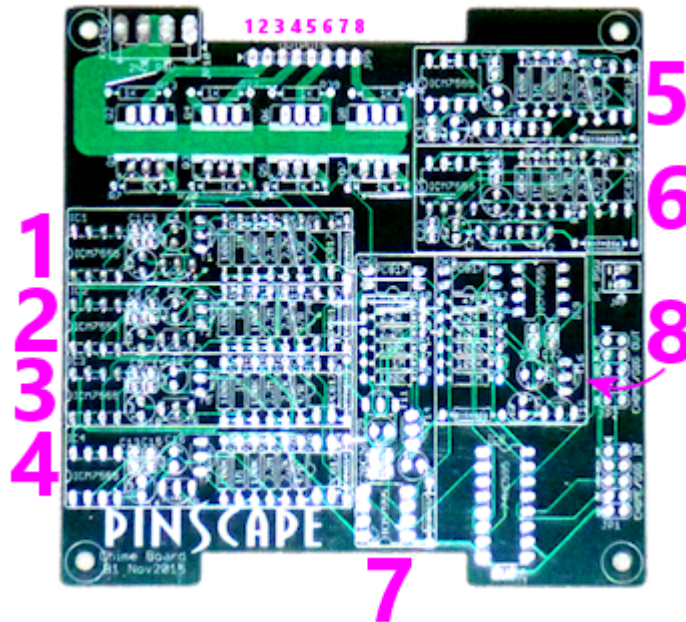
- R3
- R4
- T7

For the definitive connections in that section, see the main board EAGLE schematic for the connection to the CHIME/DIGITAL OUT header (JP5) pin 7 (ENA).

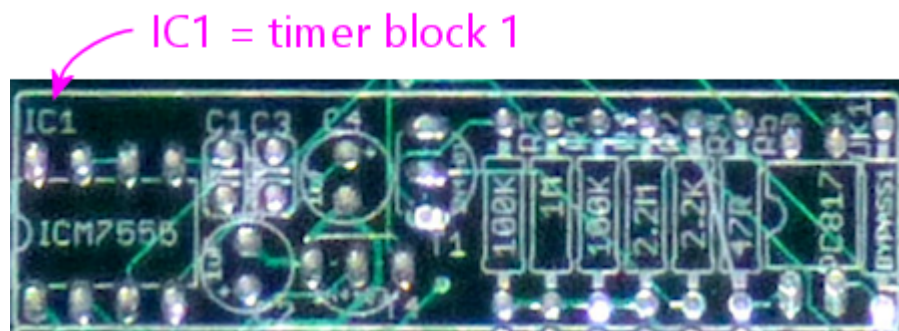
Chime board ports: some work, some don't

If some individual chime board ports are working and others aren't, the data cabling and the 74HC595 wiring must be sound, so the problem lies in the timer circuit or MOSFET for the individual non-working outputs. These are fairly complex circuits, and fortunately they don't seem to give anyone trouble, so I'm not going to go through a complete tracing process for them here.

If you do need to debug one of these circuits, note how each output's timer circuit is grouped together into a little rectangular section of the circuit board, marked on the top of the board with a box around it:



The numbers in the diagram show the correspondence between the output port pins and the timer block boxes. So if output pin 3 isn't working, for example, find the box marked 3 on the diagram, and focus on the parts within that box. You can also identify which block connects to which output by looking for the ICM7555 IC chip within the block. These are numbered the same as the outputs: IC1 is in timer block 1 for output 1, IC2 is in timer block 2 for output 2, etc.

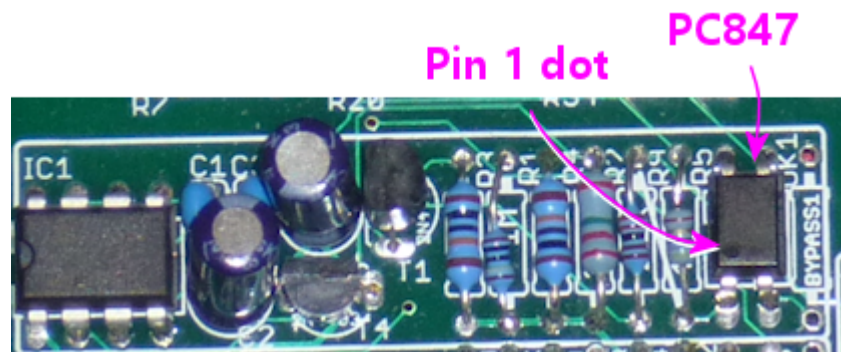


Given the number of parts and complexity of the network, it's probably more trouble than it's worth to trace individual connections and check continuity. Instead, I'd just inspect all of the solder joints within the bad output's timer block carefully, and if you can't find a visually apparent bad one, re-solder them all. Do a live test.

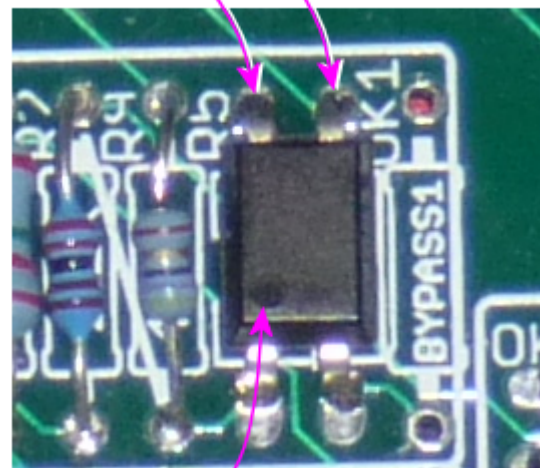
If that doesn't solve it, try the following live test. As always with live tests, be really careful not to short any pins besides the ones we're going to test.

- Find the PC847 chip within the block

- Identify pin 1 (by the dot on the chip)
- Short together the two pins shown below



Short these pins



Pin 1 dot

That test will bypass the timer block and simply activate the MOSFET. If the output turns on when you do this, the MOSFET is good, and something is wrong in the timer block. If the output doesn't turn on, the MOSFET is probably bad. To identify the MOSFET in this case, look for the "Q" label that matches the output port number: port 1 is MOSFET Q1, port 2 is Q2, etc.

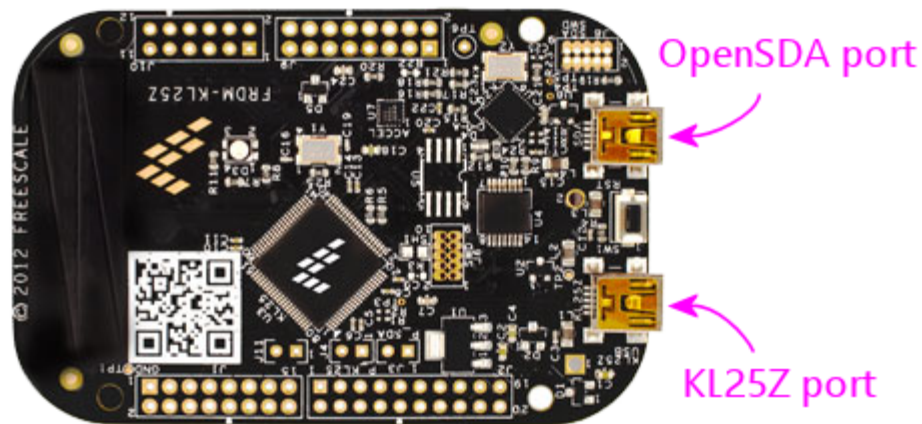
98. Connecting the Expansion Boards

Before connecting or disconnecting any of the expansion board cabling, you should have the power off to the PC and all of the power supplies. The same applies for seating and removing the KL25Z in the main board sockets.

Attach the KL25Z

The KL25Z simply fits into the sockets on the main board. It only fits in one way, since the sockets are asymmetrical.

In addition, connect the "KL25Z port" to the PC via a USB cable.



The "KL25Z port" is the one that Pinscape uses for all of its joystick emulation, keyboard emulation, and output controller features. The "OpenSDA port" is only needed when downloading new firmware onto the KL25Z, so you don't have to leave it plugged in all the time. But it's fine to do so - it doesn't interfere in any way with normal controller operations. I leave it plugged in on my cab so that I don't have to mess around with connecting cables when I want to update the firmware; it's always plugged in and ready to go with updates.

Connect power cables

All of the expansion boards have **two** power input: one labeled **PC PSU** and one labeled **2ND PSU**. Both of these inputs must be connected to power in order for the boards to operate.

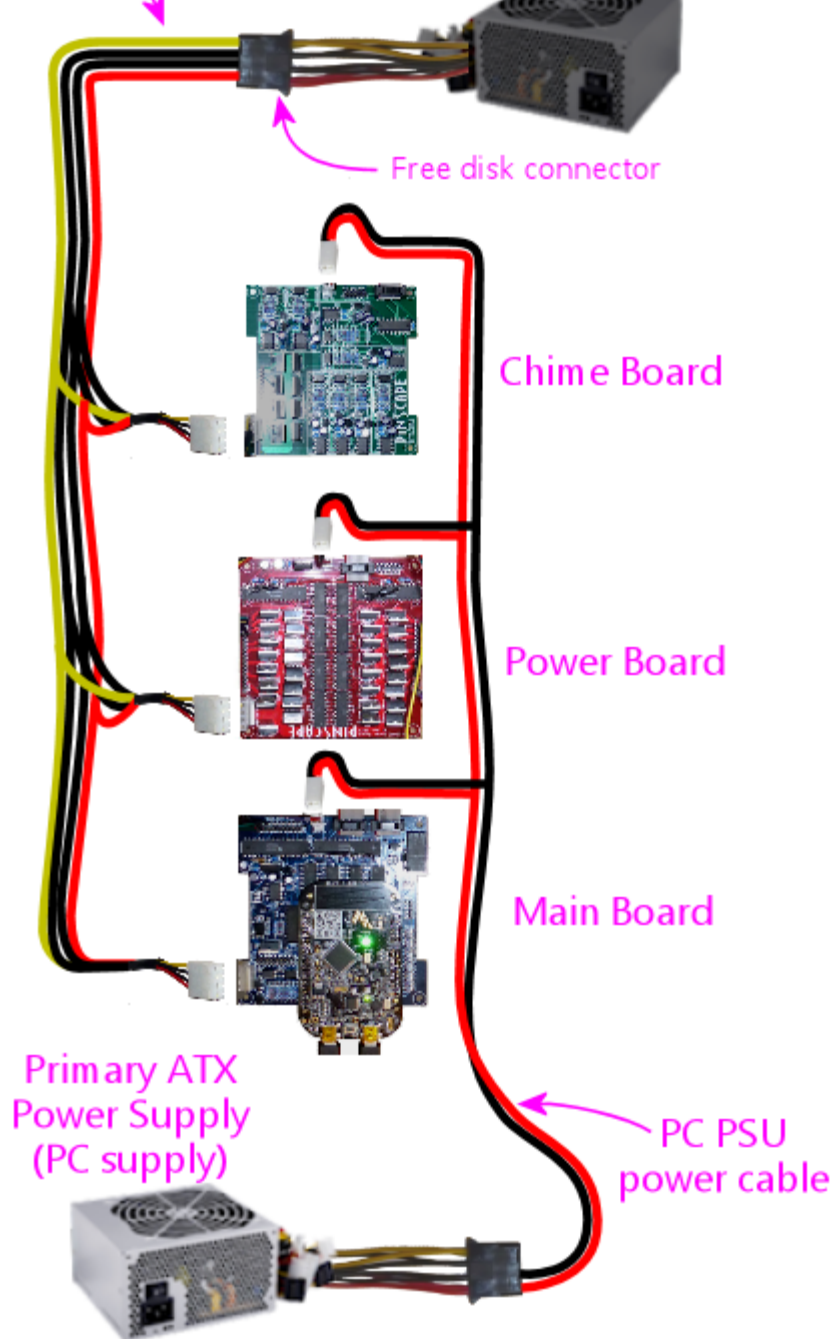
The connectors on the boards use special cables that you can build yourself. See Chapter 95, Expansion Board Power Cables for full instructions.

The **PC PSU** connector is meant to connect to the ATX power supply that's powering your PC motherboard. You should connect the cable for this connector to a free disk power plug on your PC's power supply.

The **2ND PSU** connector is meant to connect to a separate, "secondary" ATX power supply that's dedicated to powering your 5V and 12V feedback devices, such as flasher LEDs, light strips, motors, fans, beacons, and possibly solenoids.

2nd PSU
power cable

Secondary ATX
Power Supply
(Feedback devices)

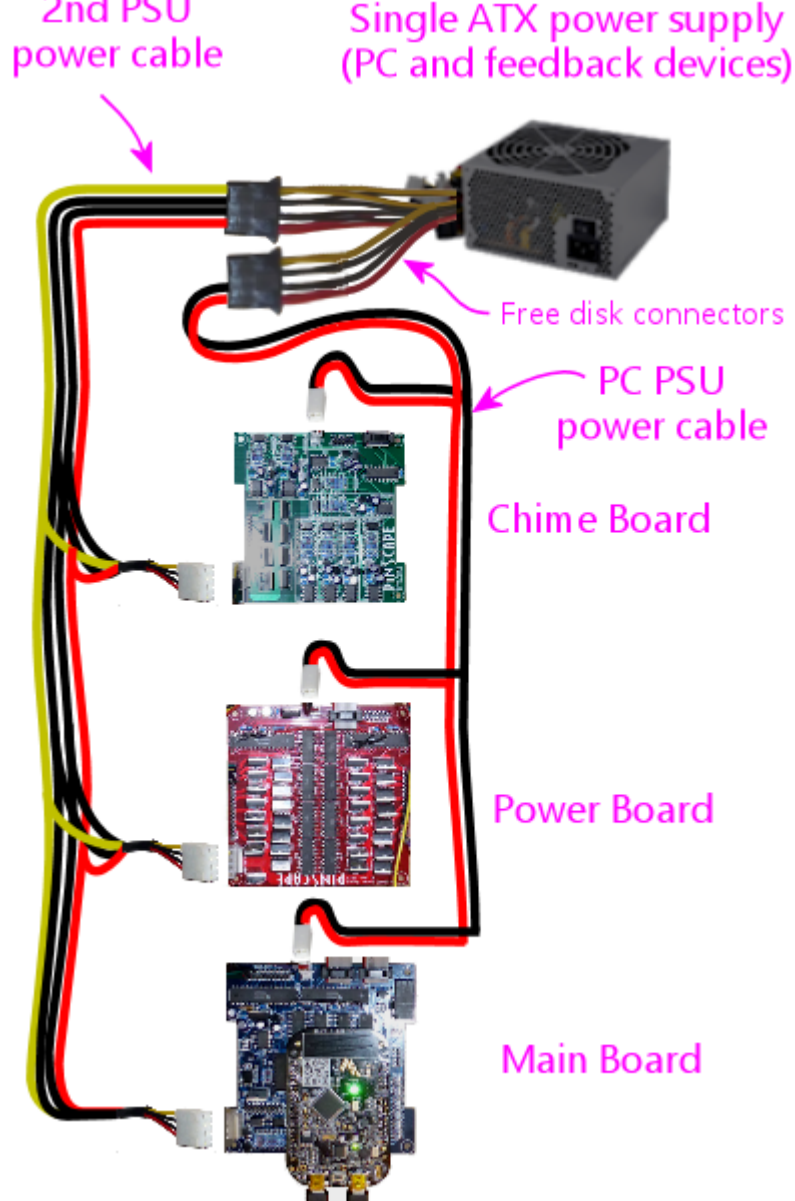


Do I really need *two* power supplies?

Not absolutely, but it's better to use two.

If you have some serious constraint that prevents using two supplies, you *can* get away with sharing a single ATX power supply for both the PC power and the feedback devices. The reason we recommend against this is that it places a lot of load on the one PSU. Some of the feedback devices draw enough power that they can momentarily overwhelm the PSU's voltage regulator, especially when switching on and off, which can make the PSU's output voltage levels fluctuate. Exposing a PC motherboard to that kind of fluctuation can cause glitches, such as USB disconnects or random Windows crashes. Using a separate power supply unit avoids placing that kind of load on the PC's supply.

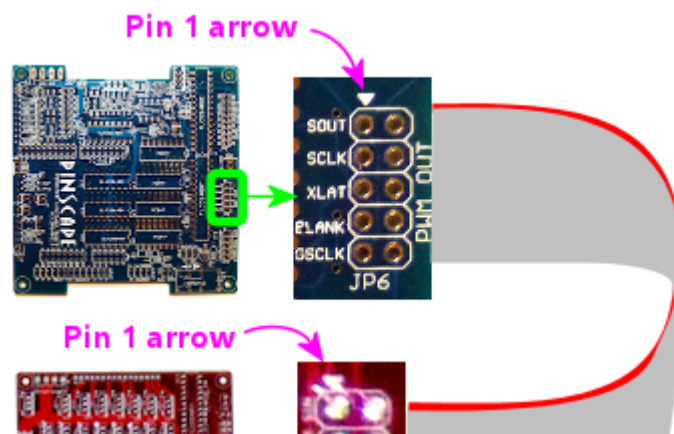
If you do decide to run everything on a single power supply, the expansion boards *still* need both connectors to be plugged in. In this case, you simply plug both connectors into the main PC power supply, rather than plugging them into separate supplies.

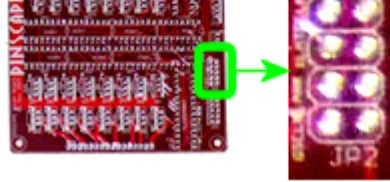


Connect the first power board to the main board

Connect the power board to the main board via a 10-pin ribbon cable. See Chapter 83, Ribbon Cables if you haven't built this cable yet. This cable connects between **JP6 (PWM OUT)** on the main board and **JP2 (PWM IN)** on the power board.

Be sure to align pin 1 on both ends of the cable. I recommend marking a red stripe along one edge down the whole length of the cable, and calling that the Pin 1 side. That makes it easier to get the orientation right at both ends, since all you have to do is align the red stripe with the "Pin 1 arrow" marked on the board at each end.

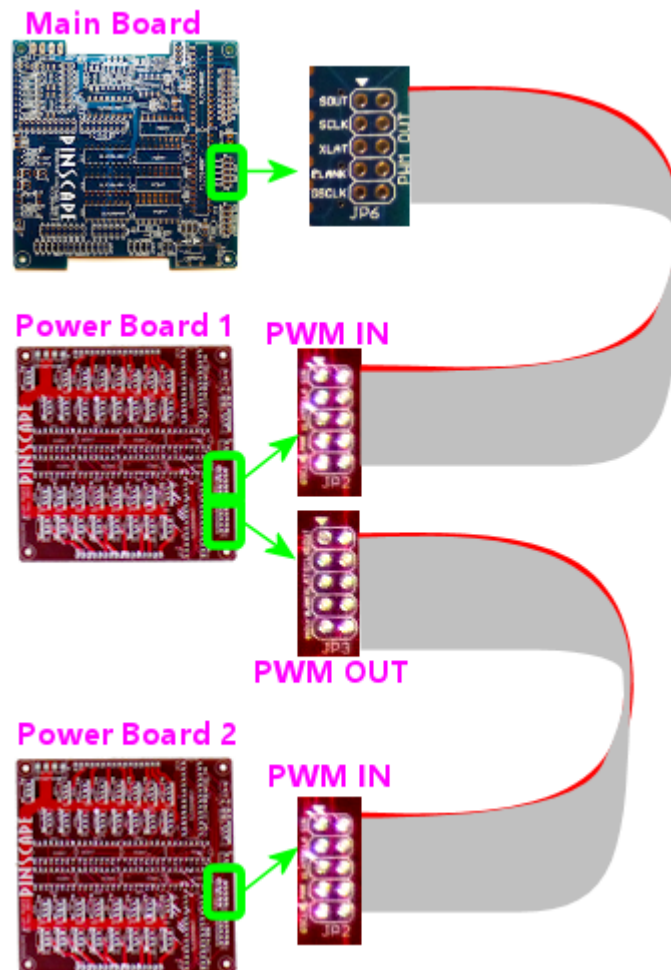




Connect additional power boards

If you're using two or more power boards, connect their data cables in a daisy chain, from the PWM OUT port of one board to the PWM IN port of the next board:

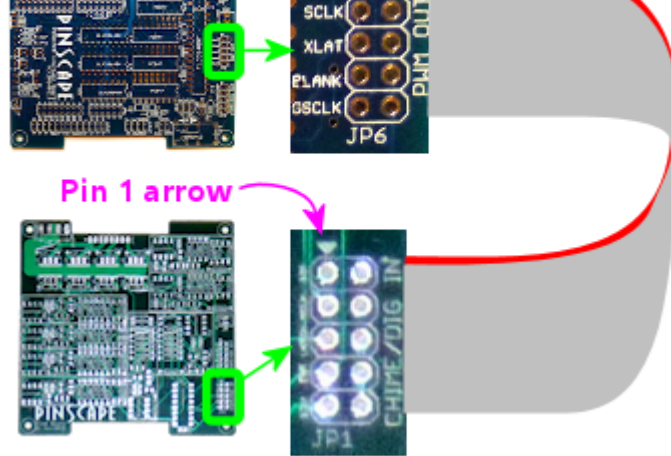
- The second board's PWM IN port connects to the first board's PWM OUT port
- The third board's PWM IN port connects to the second board's PWM OUT port
- And so on



Connect the first chime board to the main board

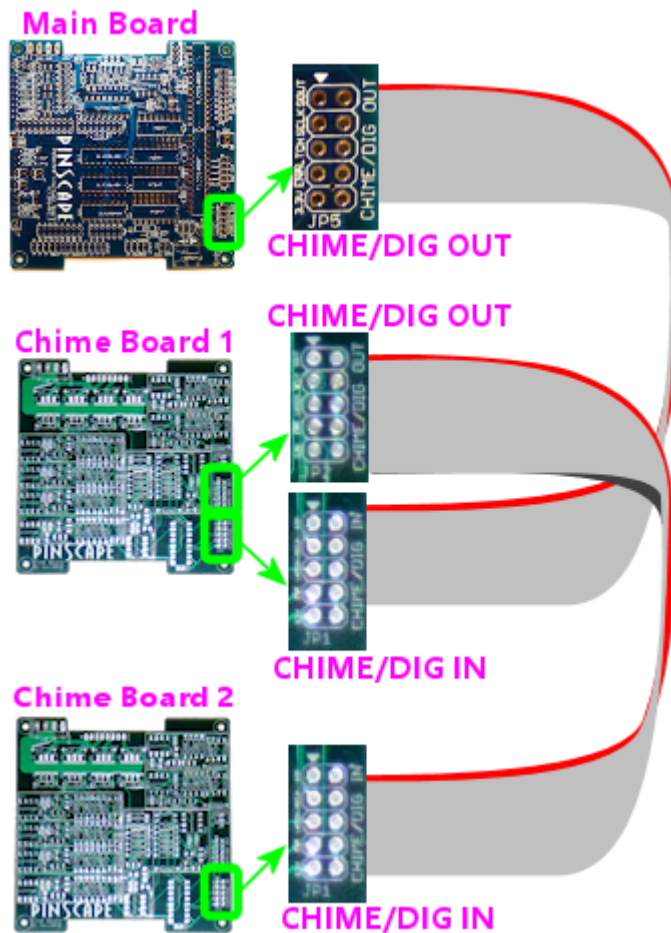
Connect the ribbon cable between the main board's CHIME/DIG OUT port (JP5) and the chime's board's CHIME/DIG IN port (JP1). As with the power board ribbon cable, make sure that the cable is aligned so that Pin 1 on the main board header connects to Pin 1 on the chime board header. I recommend marking a red stripe on one edge for the whole length of the ribbon cable to mark the Pin 1 side, to make it easier to get the orientation right when plugging it in.





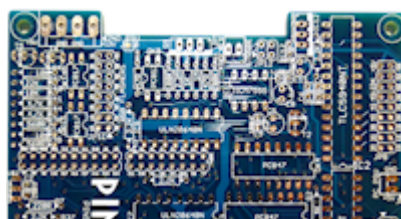
Connect additional chime boards

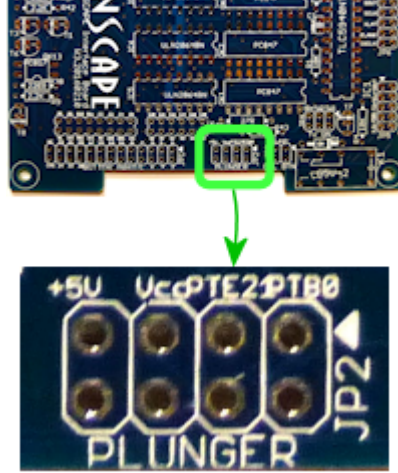
The chime boards connect in a daisy chain, like the power boards. In this case, you connect the CHIME/DIG OUT port of one board to the CHIME/DIG IN port of the next board.



Connect the plunger sensor

The plunger sensor connects to the main board pin header marked **PLUNGER (JP2)**.



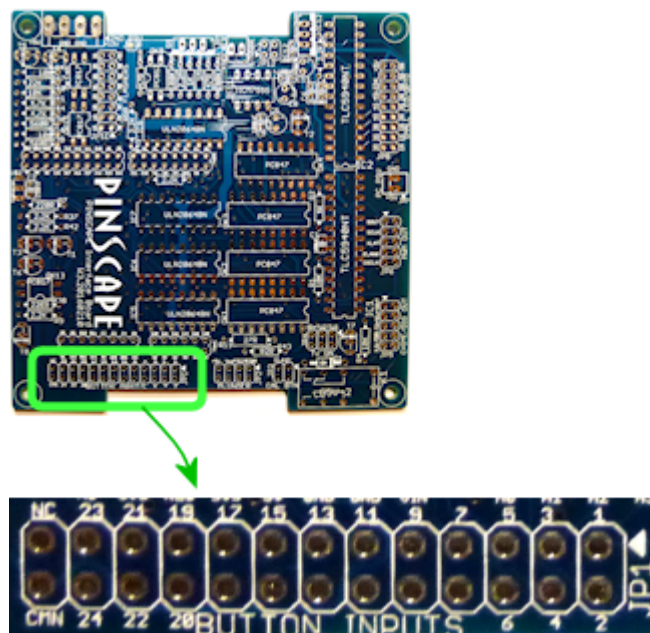


The easiest and best way to connect to this header is using a ribbon cable. There's a separate chapter for each type of plunger sensor, and each contains instructions for building the ribbon cable connector for that sensor. Also see Chapter 83, Ribbon Cables for general instructions on building ribbon cables.

When connecting the plunger sensor connector, take care to line up the **Pin 1** connection on the plunger sensor with the Pin 1 arrow marked on the main board.

Connect button inputs

The buttons all connect to the header on the main board marked **BUTTON INPUT (JP1)**.



The numbered pins on this connector correspond to individual button inputs. The pin marked CMN is the "Common" pin.

Physically, pin cab buttons are typically either microswitches (such as the SuzoHapp pushbuttons most people use for the front panel buttons, such as Start and Exit) or leaf switches (usually used for flipper buttons). In either case, the part you wire to the expansion board is a "Normally Open" switch - a switch with two terminals that come into contact when you push the button. To wire a button to the expansion board, then, you connect one terminal of the switch to one of the numbered button input pins, and you connect the other terminal to the CMN (Common) pin. All of the

buttons connect to the Common pin (thus the name!), but each button gets its own individual numbered input pin.

Full instructions for connecting the buttons can be found in Chapter 110, Pinscape Button Inputs, and much more information about buttons in general (a list of all of the common pin cab buttons, what to buy, how to install buttons in the cabinet) is in Chapter 34, Cabinet Buttons.

The pin header is designed to mate with a 26-pin crimp housing (listed in the parts list). For help assembling crimp pin housings, see Chapter 82, Crimp Pins.

Some advice on assembling the crimp pin housing for this connector: I'd start with an empty housing (without any wires installed yet), and gradually install the wires as you install the buttons themselves. One nice thing about these crimp pin housings is that you can insert pins one at a time, so there's no need to install all of the pins at once. Each time you install a button, run the wiring from the button back to the expansion board, crimp a pin on the end of the wire, and insert the pin into the housing.

The one pin you probably will want to install up front is the Common wire, since all of the buttons will need to connect to that. You might want to connect that to a terminal block or something similar, so that you can easily add more wires that connect to this same common point as you add more buttons.

TV ON connections

See Chapter 114, TV ON Switch for details on connecting the TV ON switch.

IR remote control connections

See Chapter 112, IR Remote Control for how to connect the IR remote control emitter.

Feedback device connections

See Chapter 48, Pinscape Outputs Setup (Expansion Boards) for how to connect feedback devices to the expansion boards.

99. Accelerometer (Nudge) Setup

The KL25Z has an on-board accelerometer, which is a sensor that measures the amount of acceleration applied to it. With a little classical Newtonian physics, we can use acceleration measurements to infer related information about the physical motion of the pin cab, which we can feed into a physics simulation like Visual Pinball so that nudging the cabinet can translate into a suitable effect on the game.

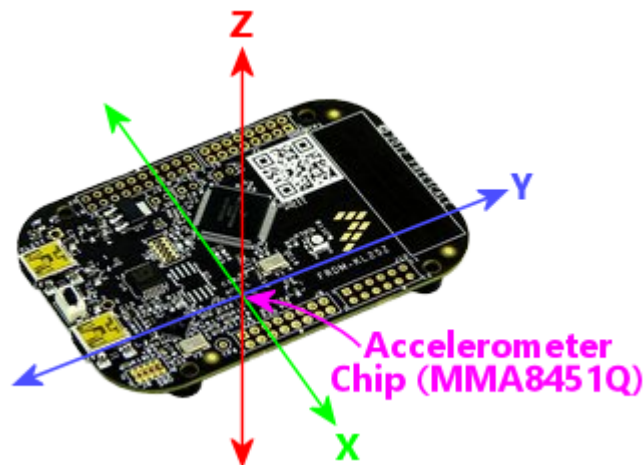
Analog acceleration sensing adds a whole new kind of "control" to pinball simulation, on par with real flipper buttons and mechanical plungers. Physical interaction with the game via subtle and not-to-subtle nudges is a huge part of real pinball that's completely lost in desktop pinball. Desktop pinball tries to make up for it with a "nudge key" that you can press, but that's not even close to the real thing; it's like trying to replace the feeling of throwing a ball with a button labeled "throw ball". The physical interaction you have with a real pinball machine is automatic, subconscious, intuitive, and the effect on the game is proportional and directional. A pin cab with an accelerometer brings that same mode of interaction to virtual pinball.

The KL25Z's accelerometer is built-in, and the Pinscape software uses it by default to send readings to the PC. So there's very little to set up on the controller side, other than physically installing the KL25Z to take best advantage of the sensor. There are some options you can set to fine-tune the input, though. And there's a small amount of work needed on the PC side to configure pinball programs to use nudge input.

This section is specifically about setting up Pinscape nudging. You might also want to take a look at Chapter 36, Nudge & Tilt, which looks at the subject more generally.

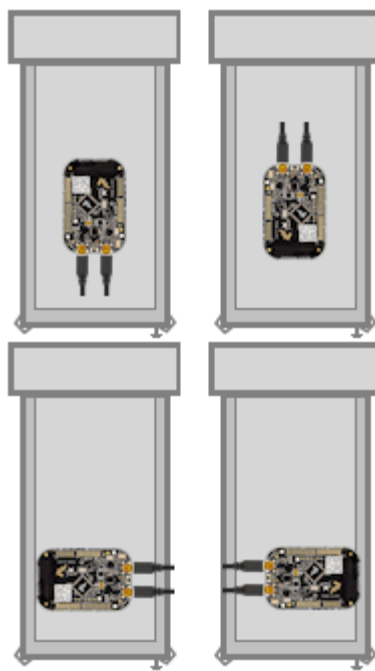
Positioning the KL25Z

The KL25Z's accelerometer is a three-axis type - it measures accelerations side-to-side, front-to-back, and up-and-down. The accelerometer chip is mounted on the KL25Z board so that the axes align with the plane of the board.



To take the best advantage of this, we want to align those native axes on the device with the axes of the cabinet, so that it's easy for the software to translate the sensor readings from the device to the motion of the cabinet. Specifically, the KL25Z board should be roughly flat on the floor, and it should be square with the front and sides of the cabinet.

"Square" with the front and sides means that the sides of the board are parallel with the corresponding cab walls. That makes any of the following orientations work (viewing the main cabinet from directly above):



So that tells us how to orient that board. The exact placement within the cabinet is a little more flexible. To a certain extent, it shouldn't matter too much where you place it, because the board will always move with the cabinet no matter where you put it. But the cabinet's motion has some subtleties because of the way it sways on the four legs, so I think it's best to locate the board close to the front, roughly centered side-to-side. That'll place it closest to where you're actually applying the nudge forces, so it should make it pick up motions in the same directions as your nudges, or at least as close as possible.

Finally it's very important to tightly attach the KL25Z to the cabinet floor. It should be secured to the cabinet so that it moves exactly as the cabinet moves. It shouldn't have any play that lets it slip around on its own; you want it to precisely track the motion of the cabinet. If you're mounting the KL25Z stadalone, attach it with a couple of screws through the mounting holes near the USB connectors, and make sure all four rubber feet are firmly planted. If you're using the expansion boards, the KL25Z plugs in via pin sockets that are a good tight fit, so you just have to make sure the main board itself is securely installed on standoffs.

To summarize:

- Place the board flat on the floor of your cab
- Top side facing up (the side with the IC chips)
- Square with the sides, in one of the four orientations shown above
- Close to the front of the cabinet
- Roughly centered side-to-side
- Mounted securely so that it moves precisely as the cabinet moves

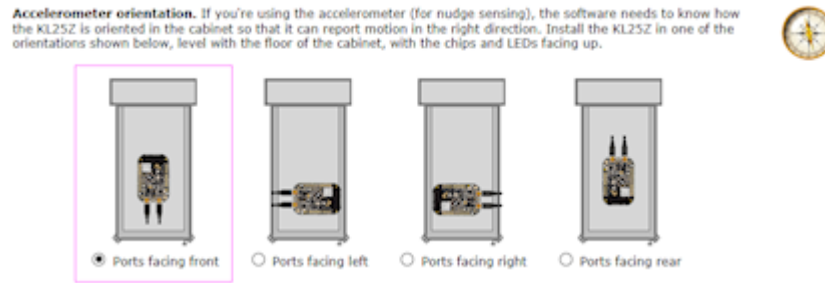
Config Tool nudge setup

The Config Tool has a few settings for the accelerometer that you can configure. To edit settings:

- Launch the Pinscape Config Tool
- Click the Settings icon for your device

- Scroll down to the **Accelerometer orientation** section.

The most important thing to set here is the device orientation. Select the radio button that matches the orientation of the physical device.



Dynamic range: this setting lets you change the maximum acceleration that the device can register. There's a trade-off: a wider dynamic range also means lower precision (the ability to distinguish small differences in degree). The KL25Z's accelerometer chip can be set to a maximum range of 1G, 2G, 4G, or 8G (meaning *N* times the acceleration due to Earth's gravity). The 1G setting has the best ability to distinguish fine shades of differences at small accelerations, down to about 1/10000 of a "G", but any actual acceleration higher than 1G (about 10 meters per second) will just read as the maximum 1G. 8G is at the other extreme: it can distinguish strong accelerations up to 8 times Earth's gravitational acceleration (which are very strong accelerations indeed), but at the low end can only distinguish to about 1/1000 of a "G".

In my opinion, the high-precision 1G setting is the best option for pin cab. A 1G acceleration is pretty strong in this context, and I don't see any practical need to distinguish nudges that are even stronger than that. We're better off with the finer gradations of subtle nudges that this setting can distinguish.

Auto-centering: If you could mount the KL25Z perfectly flat in your cabinet, it would read "zero" on the X and Y axes all the time naturally. If you have even a slight tilt, though, Earth's gravity will register as a slight constant pull on these axes. Since this is practically impossible to eliminate in the physical mounting, the Pinscape software compensates for it by subtracting out any constant acceleration on either axis. This compensation is called auto-centering.

The auto-centering option is enabled by default, with a default interval of 5 seconds. That means that any time the accelerometer hasn't been registering any significant motion for 5 seconds, the firmware will take the average reading over that period to be the true rest position of the physical device.

You can change this to disable centering entirely, or to change to a longer or shorter period. If you disable auto centering, you can use the joystick viewer in the Config Tool to manually re-zero the readings at any time.

I'd recommend leaving auto-centering enabled. The true physical orientation in space of your pin cab will inevitably change slightly over time, as a natural consequence of your pin cab shifting on its footings. The KL25Z accelerometer is so sensitive that it'll pick up these slight shifts. If you don't allow the software to compensate by auto-centering frequently, you'll see a constant small bias in the readings, as though you were constantly nudging the machine slightly. The constant compensation provided by the auto-centering software keeps these biases from developing.

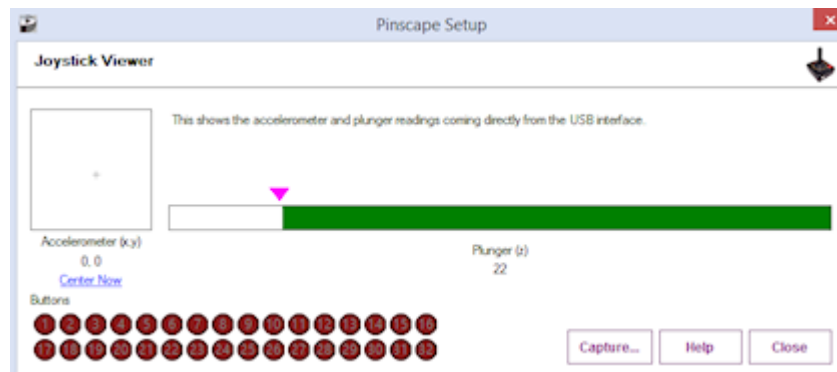
Testing nudge input

The Config Tool has a built-in viewer/tester for the joystick interface.

- Launch the Pinscape Config Tool
- Click the Joystick icon for your device



That brings up the joystick viewer window:



This lets you view the information that the device is sending to Windows, to verify that the USB joystick interface is working properly and that the accelerometer, plunger, and button inputs are working as expected.

The little diagram at the left labeled "Accelerometer" shows you the current nudge readings. The little crosshairs in the middle represents the accelerometer reading in 2D space; up/down corresponds to the front-to-back direction on your cabinet, and left/right corresponds to left/right on your cabinet.

Check the orientation: When you give your cabinet a push from any direction, you should see the crosshairs jump briefly in the same direction as the force you applied. Push the cabinet from the front, and the crosshairs should briefly deflect upwards; push from the left, and the crosshairs should deflect to the right.

If the crosshairs move in the wrong direction when you nudge the cabinet, go back and check the device orientation setting to make sure it matches the way you positioned the KL25Z in the cabinet.

Button inputs: The red circles at the bottom represent the **joystick** buttons. These don't represent the physical button inputs on your KL25Z - they're strictly the joystick button presses that Windows sees through the joystick device interface. If you programmed a button to send a keyboard key instead of a joystick button, you won't see anything light up in the joystick viewer when you press the button.

Windows calibration = bad

Windows has its own joystick device calibration procedure, which you get to via the Windows control panel called "Set up USB game controllers". Don't use it!

Everyone always wants to run this. They see the calibration option in Windows and think it must be there to help. It *is* there to help, but only for *real joystick* devices. It's a disaster to use with nudge/plunger devices, because they're not anything like

real joysticks. Nudge/plunger devices only *pretend* to be joysticks so that they don't need separate device drivers.

If you accidentally ran the Windows calibration before you read this warning (everyone does!), you'll need to delete the Windows calibration. The Windows calibration will screw up the Pinscape readings and make your nudge and plunger inputs act erratically. Fortunately, they made it pretty easy to reset the unwanted calibration data:

- Open the "Set up USB game controllers" control panel (press Windows+R, type **joy.cpl**, press Enter)
- Select the Pinscape device
- Go to the Settings tab
- Click Reset to Defaults

Visual Pinball setup

VP can use accelerometer input for simulated nudging. This isn't enabled by default; you have to adjust some settings in VP to get it working.

For instructions, see "How to configure VP for an accelerometer" in Chapter 36, Nudge & Tilt.

FX2/FX3 setup

Pinball FX2 and FX3 can also simulate nudging using an accelerometer, but they don't use the joystick interface that Pinscape provides. Instead, they require input through the XBox controller interface.

To bridge the gap, there's a program called **x360ce** that can make a joystick device emulate an XBox controller. That can reportedly be used to make Pinscape nudging work in FX2/FX3.

I don't use this in my own system, so I don't have any details about how to set it up. If anyone wants to write up instructions, I'll be happy to include them here.

Conflicts with other joystick devices

Windows is happy to let you attach multiple joysticks to your system at the same time, and doesn't have any trouble telling them apart. However, not all applications are so accommodating. Some people have run into conflicts between the Pinscape device and other joysticks or joystick-emulating devices.

There are two main ways to deal with conflicts:

- Temporarily disable one of the devices when running applications where the conflicts occur
- Switch one or the other device to use different "axis" assignment for their data input

These techniques are described in more detail below.

Temporarily disabling a device

Many of the Microsoft software development kits (SDKs) include a tool called

devcon, which lets you temporarily enable and disable individual devices. This can be a good way to deal with a conflicts with another device that you only use with certain applications. That is, you simply disable the other device when you're not using the specific applications it's needed for.

For example, one person on the forums reported a conflict with a special arcade controller device he only uses with certain video game programs. The arcade device was creating a conflict with the Pinscape device in Visual Pinball. But he didn't actually use the arcade device with Visual Pinball, so the simple solution was to disable it when Visual Pinball is running, eliminating the conflict in VP.

Installing devcon: The program comes with the Windows SDK, which you can download from the Microsoft site separately or as part of Visual Studio. You might also be able to find a plain devcon download with a Web search, although I don't think Microsoft makes it officially available separately from the SDKs.

Identifying the device: To disable a device with devcon, you need to know its USB ID. devcon itself has a way to list devices and their IDs. From a CMD prompt, go to the folder where you installed devcon.exe and run this command:

```
devcon listclass HIDClass
```

That will show you a list of devices with their cryptic internal ID strings on the left, and a device description on the right. The only problem is that the device descriptions aren't very specific; they're mostly vague things like "HID-compliant game controller". I think the easiest way to figure out which one corresponds to your device is the differential approach:

- Plug the device in
- Run the **devcon listclass HIDClass** command
- Unplug the device
- Run **devcon listclass HIDClass** again
- Compare the lists - the item(s) missing from the second list represent the device you just unplugged

Disabling the device: Once you've figured out which cryptic ID string corresponds to your device, you can disable it as follows:

- Open a CMD prompt window in **Run as Administrator** mode (necessary because we're messing with device drivers)
- Go to the folder where you installed devcon.exe
- Type **devcon disable "@xxxx"**, where **xxxx** is the full ID string from the "listclass" list

Re-enabling the device: Same procedure as above, but replace the **devcon disable** command with **devcon enable**.

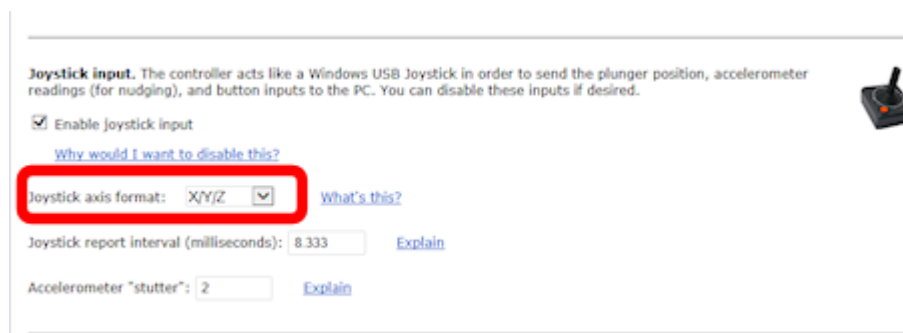
Automating it: You can automate the disable-enable procedure by putting it into a .BAT script. However, keep in mind that you'll have to run the .BAT script with "Run as Administrator" mode.

Axis settings

Another way to deal with conflicts is to tell Pinscape to report its input on a different set of axes than it normally uses. This can be useful if you're getting conflicting readings on the two devices in a program that can't distinguish between different joysticks, such as Visual Pinball.

Most joystick devices send their input to Windows via the main joystick position axes, known as the X and Y axes. That's how Pinscape normally sends its accelerometer inputs. There's also a third standard joystick axis, called the Z axis, that Pinscape uses for plunger input.

Pinscape has an option to switch its inputs from using the X-Y-Z axes to a separate set of axes, called the "R" axes - Rx, Ry, Rz. The axis assignment change can be made in the Config Tool's Settings page, under the Joystick section:



In case you're wondering, the "R" stands for rotation. In a real joystick, the R axes are used to report twisting motions on the stick. For Pinscape purposes, it doesn't really matter what the "R" axes were designed for; we just use them as an alternative way to send the same inputs.

The Rx-Ry-Rz setting can work around conflicts in VP with other joysticks that use the X-Y-Z axes. VP can't distinguish between joysticks, but it *can* distinguish the different axes.

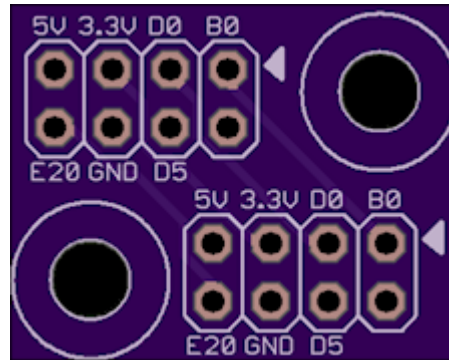
If you do change Pinscape to report readings on Rx/Ry/Rz, you have to update the joystick input settings in VP and other programs to match. VP will normally look for input on the X/Y/Z axes, so if you change to Rx/Ry/Rz, you have to tell VP to look there instead.

The downside of using Rx/Ry/Rz axes is that some pinball programs are either hard-coded to use the Z axis for the plunger, or can't use the "R" axes at all. Future Pinball can't use the R axes at all, for example. You'd lose compatibility with such programs if you moved Pinscape to Rx/Ry/Rz. So I always recommend using the default X/Y/Z axis settings whenever possible, and only changing to the "R" axes as a last resort when you have a conflict with another device that you can't resolve in any other way.

Note that Pinscape only lets you change to the "R" axes as a group. Both the plunger and nudge inputs have to be on the same axis type.

100. Plunger Sensor Breakout Board

If you're using Pinscape on a standalone KL25Z - **without** the expansion boards - this section shows how to create a little circuit board to interface between the plunger sensors and the KL25Z. This gives you a place to plug in the standard connector that we use for all of our plunger sensors.



You can skip this section if you're using the expansion boards, because they already come with a dedicated plunger connector built-in.

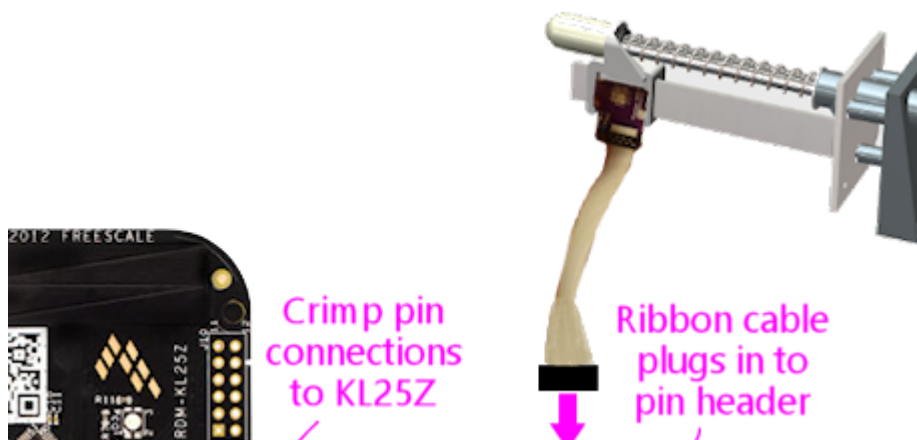
Why do you need the breakout board?

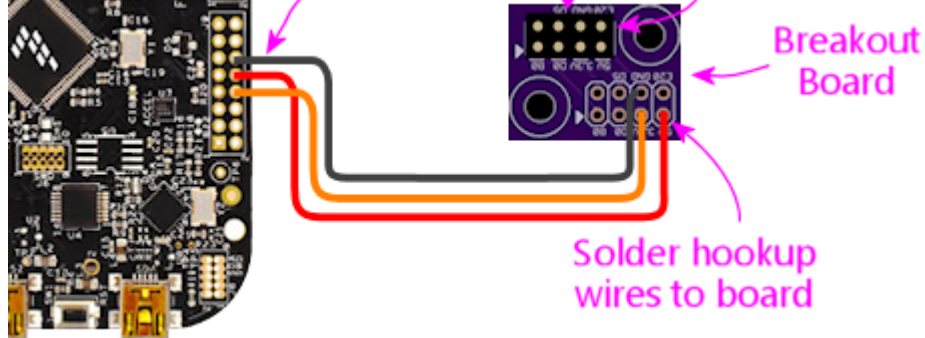
You don't *need* it, but if you're using a standalone KL25Z, the breakout board makes the connection to the plunger sensor much more convenient and tidy.

Each plunger sensor has to be connected to a set of header pins on the KL25Z. The problem is that the pins needed for any given sensor type aren't grouped together physically. They're scattered around the KL25Z pin headers. It would be nice if they could be grouped together, because then you could just plug in a small header onto the appropriate group of pins. But we can't group them, since the KL25Z's pin header layout wasn't up to us to decide.

The breakout board solves this problem by serving as an interface between the scattered pins on the KL25Z and the grouped pins on the plunger sensor cable.

- On the KL25Z, you connect hookup wires (via crimp pins) to the scattered plunger pins on the KL25Z pin headers
- These wires connect to the breakout board
- The breakout board internally routes these wires to a single 8-pin header for the plunger sensor cable
- The plunger sensor cable plugs into this 8-pin header





The 8-pin header matches the layout of the plunger header on the Pinscape expansion boards. This lets you build the plunger sensor cabling the same way as you'd build it for the expansion boards. It also means that you can easily transfer your sensor to the expansion boards if you decide to switch to those in the future.

If you're using the expansion boards, you don't need the breakout board, since it has the equivalent connect already built in.

How to order the board

You can order one from OSH Park for about \$2.50 delivered. Actually, that \$2.50 will buy you *three* copies of the board (that's their minimum order size), so you'll have a couple of extras to share, or just in case.

To order from OSH Park:

- Download the board design: mjrnet.org/pinscape/downloads/plunger-breakout-board.zip
- Unzip that and pull out the **.brd** (board design) file
- Go to oshpark.com
- Follow the instructions on their site to upload the .brd file above
- Click through the ordering process

Parts

- The board (see above)
- One 2x4-pin 0.1" pin header, such as Harwin M20-9760446, or a 2x4 section of a breakaway header (see Chapter 81, 0.1" Pin Headers)

How to assemble

Step 1: Solder the pin header. Fit the 2x4 pin header to the top of the board in one of the matching outlines (either one you prefer), with the short end of the pins facing the board. Feed the pins through the holes, making sure the plastic base is seated flush with the top of the board. Solder the pins on the bottom side. Solder all eight pins.

Step 2: Determine where it'll be installed. I recommend installing the breakout board close to the KL25Z, to keep the wiring run short. Keep it within about a foot of the KL25Z if possible.

Step 3: Cut wires. Cut a set of wires as needed for your sensor. See the sensor chapter for the specific wires you need. Cut each wire to the length needed to

connect between the KL25Z and the breakout board, according to where you plan to situate the two boards.

Step 4: Strip wire insulation. Strip about 3/16" of insulation from the end of each wire.

Step 5: Attach crimp pins. Attach a crimp pin to **one end** of each wire. Use the type of crimp pin that matches the wire housings you're using to connect to the KL25Z, because these pins will fit into those housings. See Chapter 82, Crimp Pins for instructions on how to attach them.

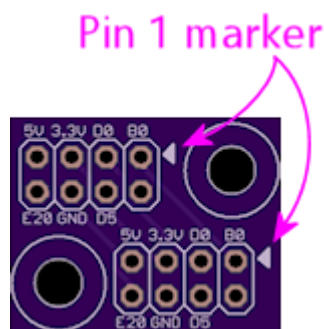
Note! If you're already using the KL25Z's 5V, 3.3V, and/or GND pins for other purposes, and you already have a crimp pin in any of those slots, you'll have to tap into your existing wire, since you obviously can't add a second crimp pin to the same slot. So instead of adding a crimp pin on any affected wires, splice into the existing wire:

- Snip the wire in two at a convenient point
- Strip a little insulation from each end of the snipped wire
- Make a three-way solder joint between the two ends of the snipped wire and the new wire to the breakout board
- Cover the exposed solder joint with electrical tape

Step 6: Solder the wires to the breakout board. Solder the bare end of each hookup wire to the appropriate slot in the breakout board - the one where you **didn't** install the pin header. Again, refer to the specific instructions for your selected sensor type for which wires to connect.

Step 7: Insert the crimp pins. Insert the crimp pins on the hookup wires into the appropriate slots on the KL25Z crimp pin wire housings. Once again, the appropriate slots depend on your sensor type.

You're ready to go. When you finish building your plunger sensor, you can simply plug in the ribbon cable to the pin header on the breakout board. Be careful to observe the "pin 1" arrow on the board - it must line up with same side of the ribbon cable that goes to the marked pin 1 on the plunger sensor side.



101. Plunger Setup (TSL1410R Optical Sensor)

The TSL1410R was the original Pinscape plunger sensor, but unfortunately, it was discontinued by the manufacturer around 2016 and is no longer available from any source I'm aware of. I haven't been able to find any other devices available with similar characteristics, either, so I can't even recommend a substitute.

The TSL1410R, when it was available, was an optical sensor - basically a camera without a lens - with all of its receptor pixels arranged in a single row about $3\frac{1}{8}$ inches long. This made it ideal as a plunger position sensor, because a standard pinball plunger has a travel of just about 3". To set this up, you positioned the sensor just above the plunger, so that the single row of pixels was parallel to the plunger's travel axis, with the pixel window facing down. You also arranged a light source, preferably a point source like a single bright LED, below the plunger, facing the sensor. In this geometry, the plunger rod cast a shadow on the sensor. To determine the plunger position, the software took snapshots of the "image" the sensor saw, and found the edge of the shadow cast by the plunger - the boundary between the bright pixels that the plunger rod wasn't blocking and the dark pixels where the plunger cast its shadow. The TSL1410R had a pixel pitch of 400 pixels per inch, and the shadow's edge position could be detected to a precision of about 3-4 pixels, so the software was able to determine the plunger's position to about 1/100 of an inch. This is about the same as the on-screen pixel pitch in an HD monitor, so it allowed the on-screen animation of the virtual plunger to very closely match the actual position of the physical plunger.

Build details

Given that this sensor is no longer available and that no substitute is available, I'm going to omit the details of how to set it up. If you're lucky enough to find one of these devices somewhere, you can still find the full setup instructions in the old "version 1" Pinscape Build Guide.

Alternatives

Fortunately, the obsolescence of the TSL1410R doesn't mean you're out of luck. There are some excellent alternatives available. The Pinscape software is "sensor agnostic" and can work with a variety of different sensor types. The best current options, in my opinion, are:

- The Chapter 105, TCD1103 optical sensor. Like the TSL1410R, this is a linear imaging sensor that takes rapid pictures of the plunger and detects its position by analyzing the images. The TCD1103 is more complex to set up than the TSL1410R was because it requires a focusing lens, as well as some extra electronics to interface with the KL25Z. But with the lens, it provides even better resolution than the TSL1410R did, and like the TSL1410R, it has no mechanical contact with the plunger, so there are no moving parts to wear over time.
- The Chapter 104, AEDR-8300 quadrature encoder. This can yield extremely precise and stable position readings (even better than the TSL1410R did), so it's a top pick for quality. However, it's rather complex to set up, and moderately expensive (about \$40 in parts). It also has the downside that the sensor is mechanically linked to the moving parts in the plunger, which can cause wear over time.
- A Chapter 102, potentiometer based sensor. Most people I've talked to who

have set these up are very happy with them. This approach is quite inexpensive (about \$6 to \$20) and really easy to set up.

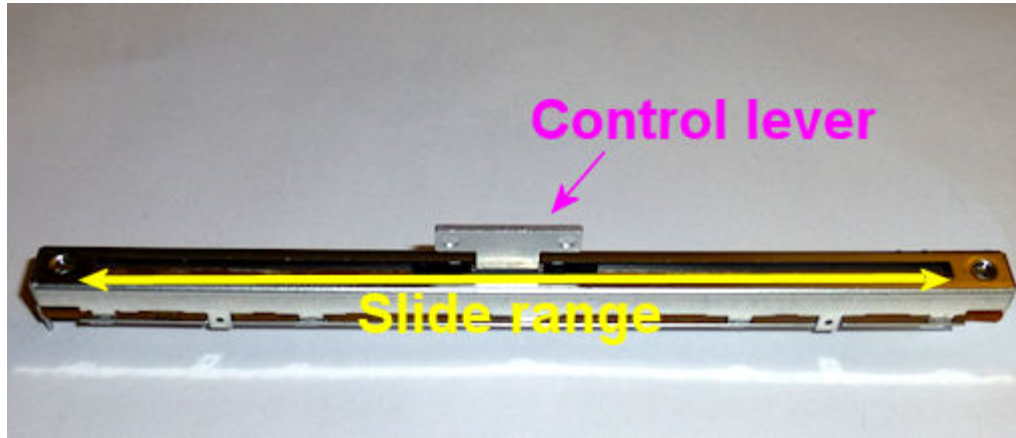
- A Chapter 103, VCNL4010 proximity sensor. This sensor works by using an IR light beam to measure the distance between the sensor and the plunger. It's not as accurate as a quadrature sensor or potentiometer, but it's respectable, and it's cheap (\$8) and very easy to set up. It's also completely non-contact, so there's no wear and tear from moving parts.

If you're very picky about the sensor quality, and you're not daunted by the extra work involved, I'd go with the AEDR-8300. For most people, though, I'd recommend giving the potentiometer a try first, since it's so easy and cheap. If you don't end up liking it, you can always upgrade to the AEDR-8300 (or maybe some even better option that comes along down the road), and I think the pot is cheap and easy enough that you won't feel like you're wasting a lot of money if you try it and end up replacing it.

There's another option as well: come up with something completely new! The Pinscape software is open-source C++, and it's specifically designed so that you can plug in new code for different sensors. (That's how it manages to support all of these unrelated sensor types already.) The plunger is represented in the code by a C++ abstract class that you can subclass with the interface logic for just about any kind of physical device. Position sensing is an interesting and surprisingly tricky problem, and I haven't come up with a perfect solution yet - all of the current options have some tradeoffs. New ideas for better sensor options would be most welcome. (See Appendix 3, Plunger Sensor Technical Notes for some ideas about future sensor experiments I'd like to try someday.)

102. Plunger Setup (Potentiometer)

The easiest type of plunger sensor to build is a sliding potentiometer. A potentiometer (often called a "pot" for short) is basically a continuously variable resistor. The sliding type has a control lever that slides back and forth in a straight line - just like a plunger. By connecting the end of the plunger rod to the potentiometer lever, we can make the lever slide back and forth in sync with the plunger. This in turn makes the potentiometer's electrical resistance change in sync with the plunger position, so by measuring the resistance, we can determine the position. The KL25Z helps us out with this by providing a built-in ADC (analog/digital converter), which can precisely measure the analog voltage on a GPIO pin.



This is one of the simplest possible position sensors, since all we have to do is connect three wires between the pot and the KL25Z (power, ground, and the variable resistance).

Here's how this works for sensing plunger position:

- Physically, you install the potentiometer in the cabinet adjacent to the plunger, fixed in place. You connect the control lever on the pot to the end of the plunger, so that the control lever moves in unison with the plunger.
- Electrically, the resistance on the pot varies according to the control lever position, so whenever you move the plunger, it changes the resistance on the pot accordingly.
- The software tracks the plunger position by monitoring the electrical resistance (actually the voltage, which is directly related to the resistance) and converting that to a position reading.

Potentiometers are inexpensive (about \$6) and easy to install. They work well, but they're analog devices, so they have a little random noise in the signal that can be visible as a small amount of jitter in the on-screen plunger position. The Pinscape software tries to minimize that by taking many readings and averaging them together, and has an adjustable "jitter" filter to calm any remaining noise in the signal so that it doesn't show up as jumpy on-screen animation.

Oak Micros's pre-built kit

No longer available as of June 2021. Oak Micros formerly sold a kit with all the parts needed for the plunger setup, including an easy-to-install mounting bracket, but announced in June 2021 that they're no longer selling their products. Here's the original announcement on vpfforums, if you want to check for any updates:

Selecting a potentiometer

The Chapter 91, parts list section has a specific slide potentiometer that's known to work well, but there are other similar devices available. You can use any slide potentiometer that meets these guidelines:

- 10 k Ω overall resistance
- Linear taper
- About 100mm travel length

Important: Pay special attention to the "linear taper" part. It's **not** a reference to the physical arrangement with the sliding control lever. It's an electrical spec that describes how the pot's resistance varies across its range. There are two main types of pots: "linear taper" and "audio taper". A linear taper means that if you plot the resistance vs. the control arm position, you get a straight line. With an audio taper, you get a logarithmic curve. It's important to get a pot with a linear resistance taper, because it gives you nice uniform distance readings across the whole sliding range. If you get an audio taper type instead, the software won't be able to convert the readings properly.

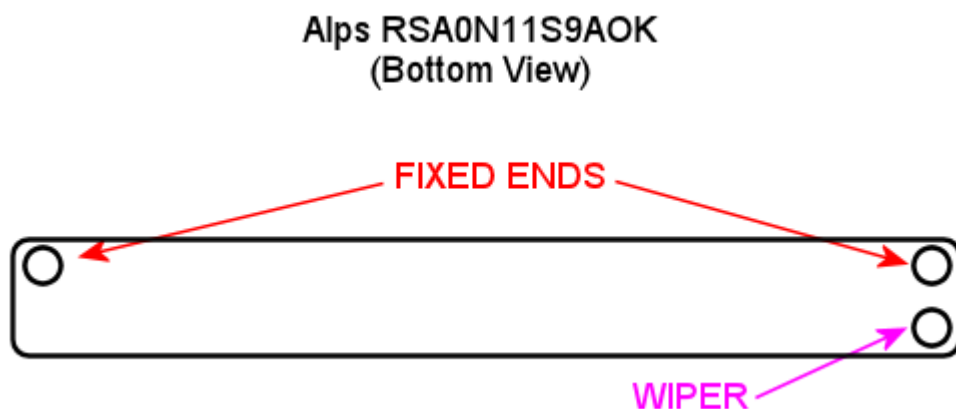
The absolute minimum length that will work is 85mm, but I'd look for something in the 100mm range, because that gives you a little leeway at each end to make sure the plunger doesn't bash the lever against the potentiometer's limits.

If you're using the expansion boards, you'll also want to pick up the connector specified in the parts list to make it easier to plug into the boards. If you're using a standalone KL25Z (without the expansion boards), see Chapter 88, KL25Z Hardware Setup for recommendations on connecting wires to the board.

Wiring

The wiring for a potentiometer is pretty simple, since there are only three wires involved. The only tricky part is identifying which terminal on the potentiometer is which, since the pins on the device itself are often unlabeled.

If you're using the recommended potentiometer from the Chapter 91, Electronic Parts List, the pins are arranged like this:



If you're using a different potentiometer, I can't tell you which pin is which, since every device has its own pin arrangement. If the device came with a diagram

showing the pins, consult that, otherwise check the vendor's Web site to see if they have a diagram.

If you can't find any information on your device, you can figure out the pin arrangement by testing with a multimeter. Follow this procedure:

- Set the meter to Ohms (Ω)
- Pick two pins to test
- Attach the leads from the multimeter to the two pins you selected
- Check the reading:
 - If it reads 10 k Ω (or whatever your device's overall resistance is, if different), and it doesn't change as you slide the control arm back and forth, you've found the two **fixed** terminals
 - If it reads something below 10 k Ω , and you see the reading changes as you slowly move the control arm back and forth, one of the pins you've selected is the **wiper** and the other is a **fixed** end

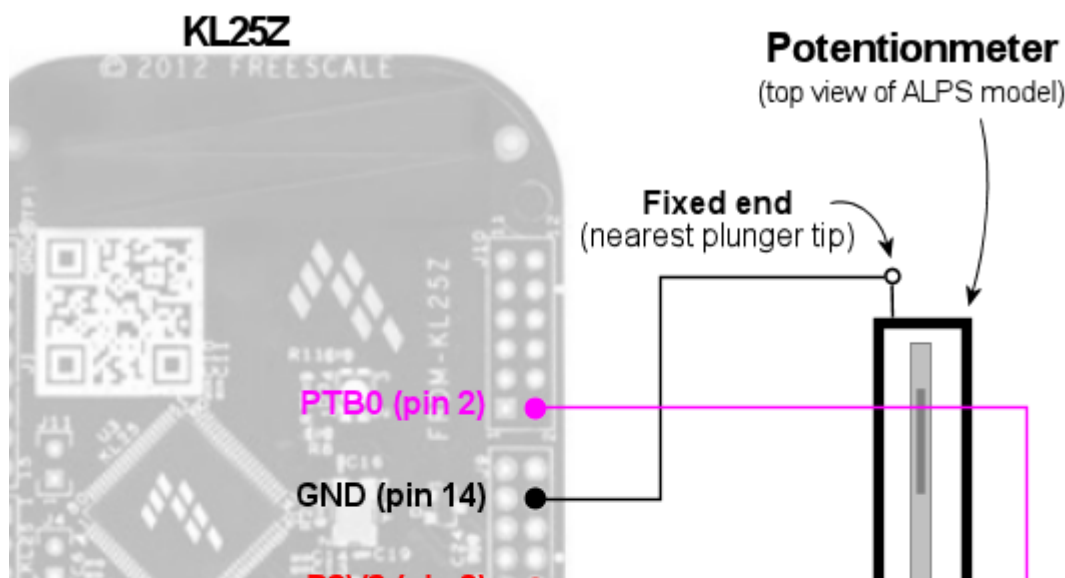
Test each pair of pins like this in turn until you hit upon the two fixed terminals. Once you know which two are the fixed ends, you know that the other one has to be the wiper by process of elimination.

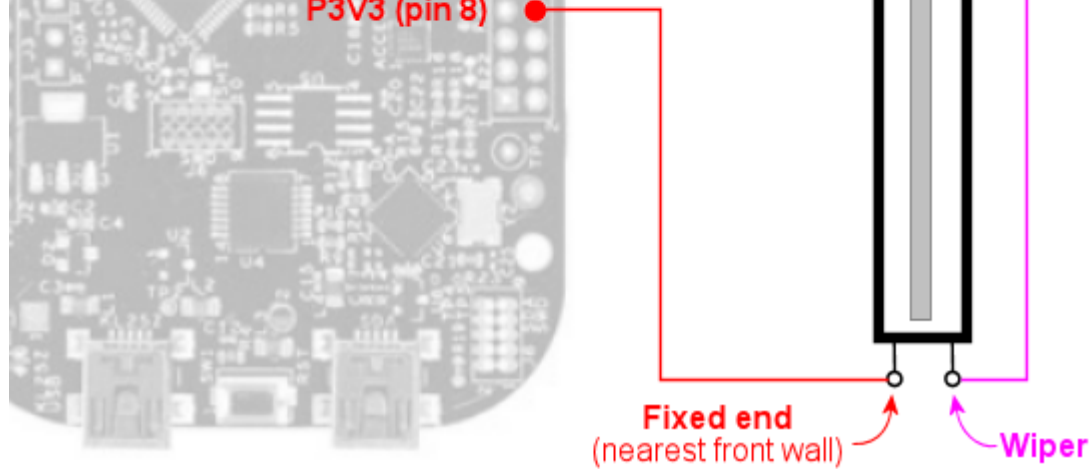
Six-pin pots

If your potentiometer has six pins, you probably bought a type that has two electrically separate potentiometers inside, both controlled together by a single control arm. In this case, there will be more combinations of pins to test, and there will be another type of reading you'll see for some of the pairs: "Infinity" or "no connection". These readings mean that the two pins you're testing are on the two separate internal pots, so there's no connection between them at all. Simply ignore these pairings, and continue testing other pairs. Proceed until you find two fixed ends and a wiper that are connected together and show readings as described above. Once you find those, you'll only need to connect those three pins to the Pinscape unit. You can leave the other three pins completely unconnected.

Standalone KL25Z wiring

Here's the schematic for wiring the stand-alone KL25Z, without the expansion boards. The two "fixed" terminals on the potentiometer are wired to 3.3V and Ground on the KL25Z, respectively. The "wiper" or "variable" terminal on the potentiometer is wired to an "analog in" port, such as PTB0.



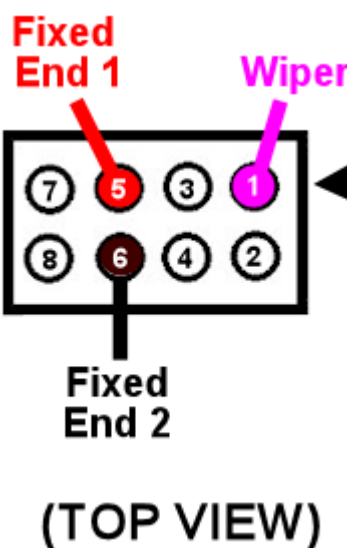


I recommend using a Chapter 100, plunger sensor breakout board to connect to the standalone KL25Z. It'll make things easier in the long run by giving you a pluggable connector between the plunger and KL25Z.

- Follow the "Expansion board wiring" instructions below to build a pluggable connector for the potentiometer itself
- Follow the instructions in Chapter 100, Plunger Sensor Breakout Board to build the breakout board
- Connect the following wires between the breakout board and the KL25Z:
 - Breakout board **B0** to KL25Z PTB0 (pin 2 on J10)
 - Breakout board **3.3V** to KL25Z P3V3 (pin 8 on J9)
 - Breakout board **GND** to KL25Z GND (pin 12 or 14 on J9)

Expansion board wiring

For the expansion board connector, build a 4x2 crimp pin housing (housing, pins). First crimp a pin to the end of each wire (see Chapter 82, Crimp Pins). Insert the pins in the housing, following the diagram below for the pin placement.



Physical installation

I'd recommend getting all of the electrical wiring to the potentiometer in place before installing the pot in the cab, since it'll be hard to access the little contact pins on the pot after installing it.

If you're using the ALPS potentiometer recommended in the parts list, you can find 3D printer plans for a mounting apparatus below, along with installation instructions.

If you're using a different potentiometer, you'll have to either modify our 3D printer plans to fit your device, or improvise something of your own. It's pretty easy to come up with an ad hoc mounting system using off-the-shelf hardware. Here are the basic requirements:

- The body of the potentiometer needs to be fixed in place in the cabinet.
- The potentiometer's control knob must be firmly anchored to the plunger rod, so that moving the plunger moves the control knob by the same amount. The attachment should have as little play as possible.
- Don't worry about the orientation of the sensor at this stage. If you get it backwards, you can simply tell the software to reverse the readings.

3D printer plans for ALPS potentiometer mounting

If you're using the ALPS potentiometer from the Chapter 91, parts list, here's a set of 3D printing plans you can use to mount it in your cabinet.

ALPS_Mounting_v6.zip

Other parts needed:

- Standard Williams/Bally plunger assembly
- M3 x 5mm to 8mm machine screws, quantity 2 (the exact length needed might vary depending on how the 3D-printed parts come out, so you might want to buy both lengths and use the one that fits best)

Fabricating the parts: The ZIP file contains five STL files. Four of the STL files are the individual pieces that make up the bracket assembly. The fifth file, **Combined Parts.stl**, contains all of the other parts joined together into a single contiguous 3D object, using the "model airplane" style where the piece are connected by little plastic struts. The combined parts file lets you manufacturer all of the parts in a single printer run.

You only need to print **either** the individual part files **or** the single combined parts file. If you're using a commercial 3D print service, I'd recommend using the combined parts file, because many of the online services charge an extra fee for each file you print, so it's cheaper to print the whole batch as a single file. If you have your own 3D printer at home, use whichever approach is more economical and easier for you.

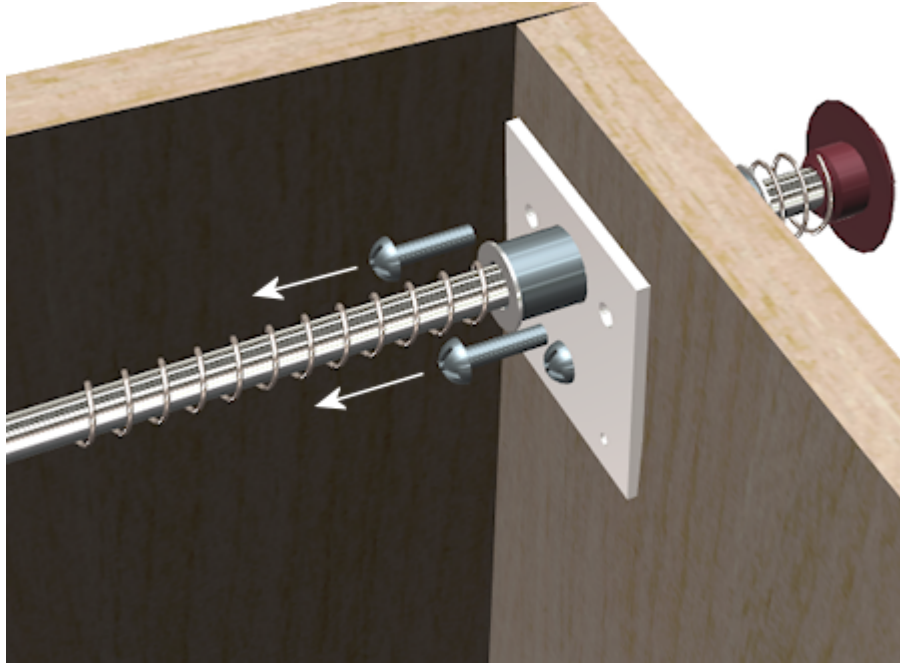
3D printing guidelines:

- The STL files use **millimeter (mm)** units
- You can use any plastic material you wish, but I'd recommend Nylon PA-11 or PA-12 if you're using an online service, as those have excellent durability
- See Chapter 4, Resources for links to online 3D print services if you don't have a 3D printer at home

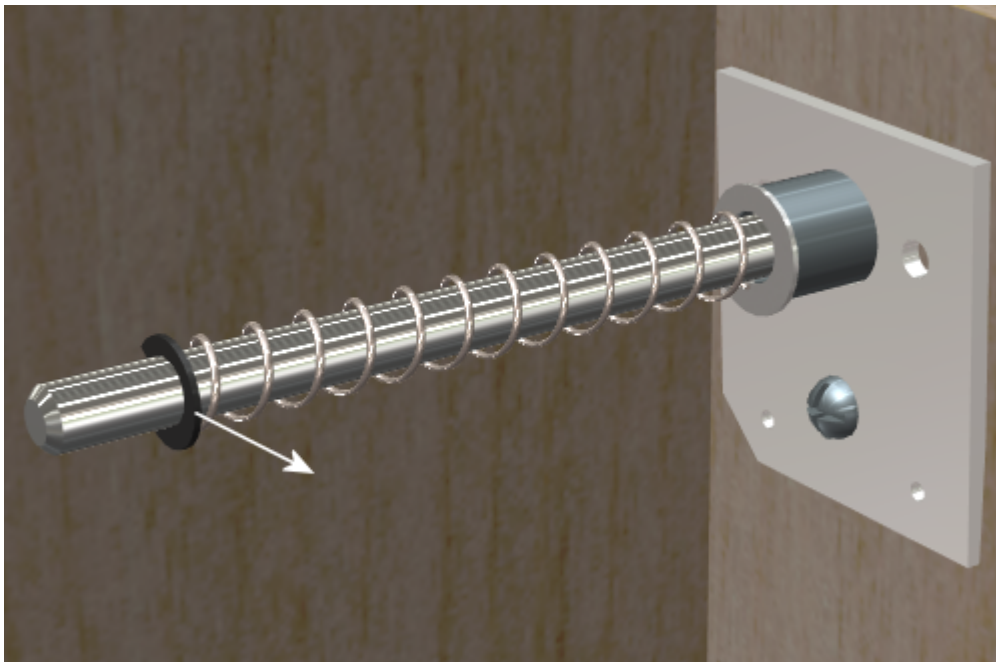
Note that my illustrations below show each part in a different color, but that's only for the sake of clarity in the illustrations. There's no need to use different colors for the actual printed parts.

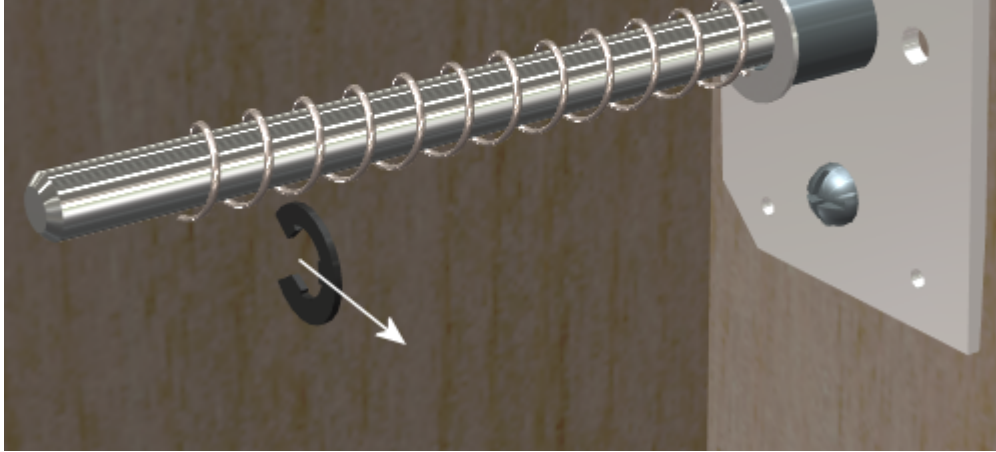
Installation: If you printed the combined parts file, separate it into the individual parts, by cutting the pieces apart at the little struts holding them together.

If you've already installed the plunger assembly in your cabinet, remove the **top two screws**. Leave the bottom screw installed. If you haven't installed the plunger assembly yet, install it now, fastening only the single bottom screw for now (leave the top two screws out).



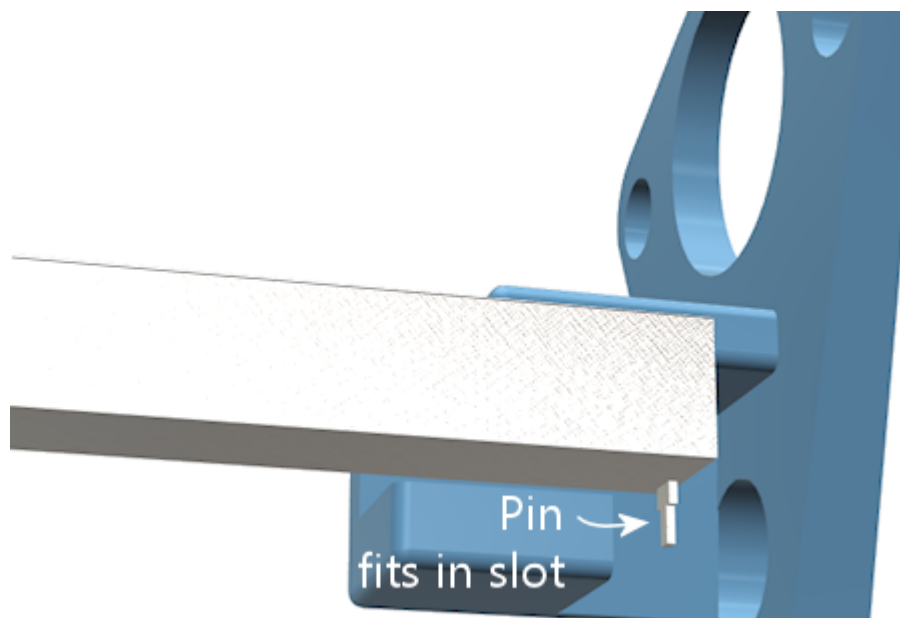
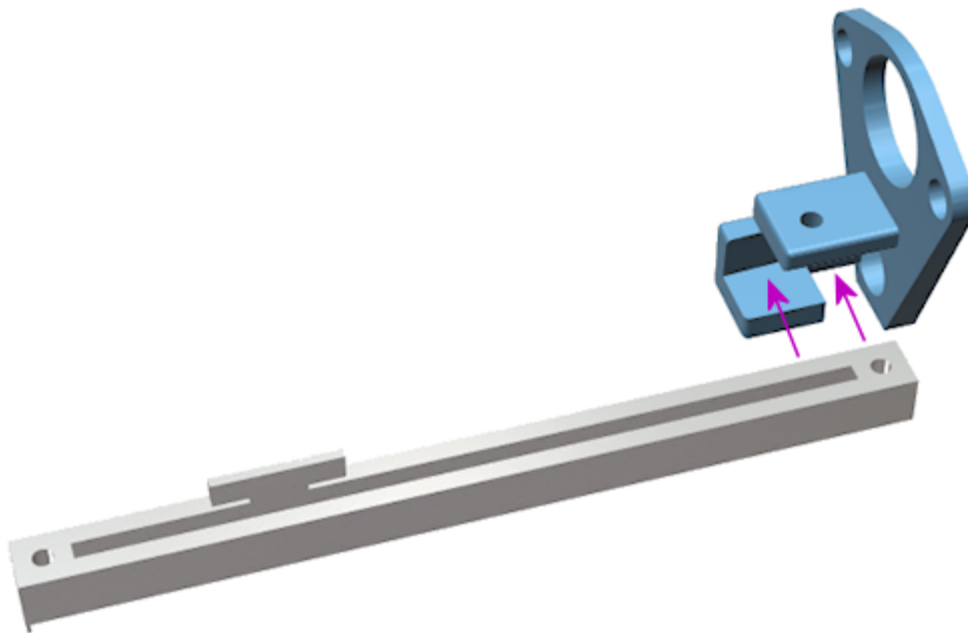
Remove the rubber tip from the end of the plunger if it's there, and remove the e-clip. The e-clip is the little semicircular metal fastener near the end of the plunger that holds the spring in place. The e-clip simply snaps out by pulling it sideways. You'll need to use needle-nose pliers to get a strong enough grip. It also helps to hold back the spring while working so that it doesn't apply extra pressure to the clip. You can leave the spring in place after removing the e-clip.





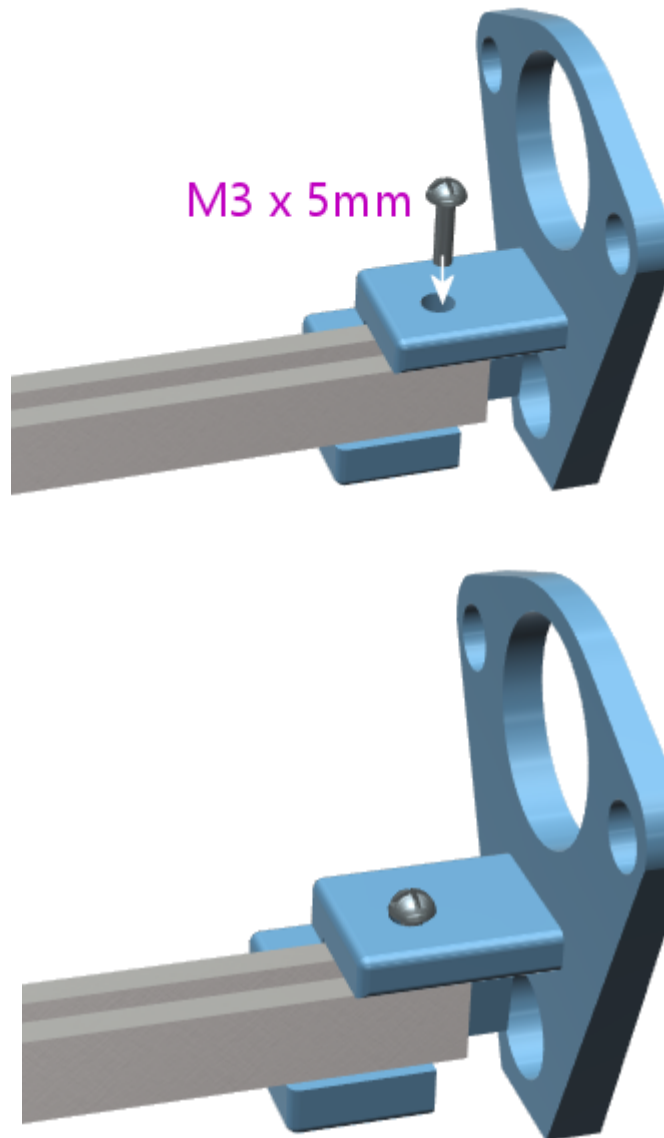
From the plastic set, find the "front bracket" piece illustrated below. Slide the potentiometer into the slot as shown. Be careful not to damage the electrical connector pin on the bottom of the pot - the pin should fit into the slot at the front.

It doesn't matter which end of the pot you call the "front" at this point. You can easily switch directions in the software after installing it, so it doesn't matter if it's "backwards" initially.



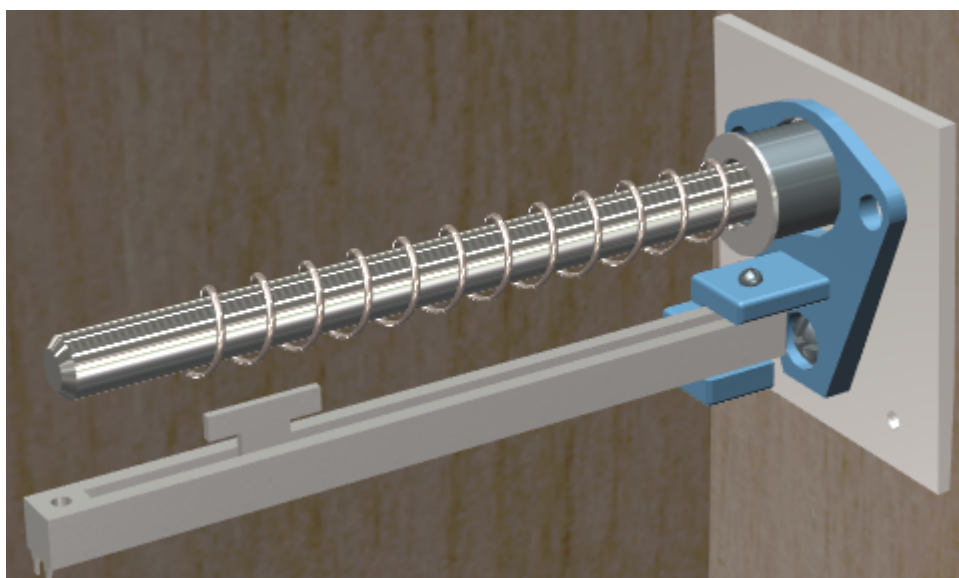
When the potentiometer is seated properly in the slot, the mounting screw hole at the end of the potentiometer should align with the screw hole in plastic piece. Install an M3 machine screw in the mounting hole to fasten the pot and bracket together. (As mentioned earlier, there can be some variation in how the 3D-printed parts fit, so the ideal screw length needed can vary; something in the 5mm to 8mm range should fit. Do a test fit with the different sizes and choose the length that works best in your setup.)

Leave the screw a little loose for now, since we'll want a little play when we finalize the alignment with the other parts later.

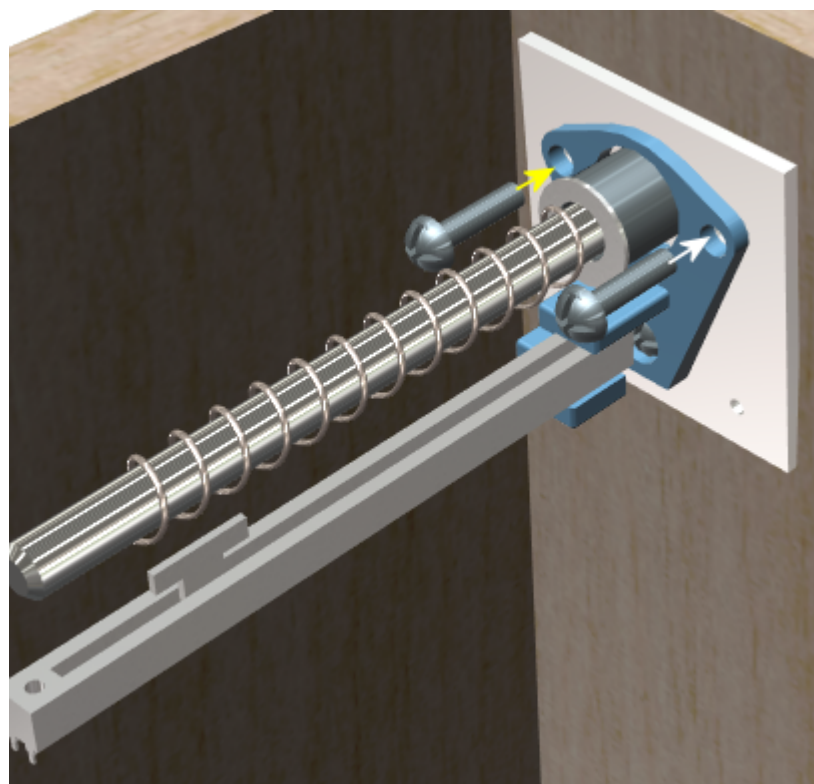


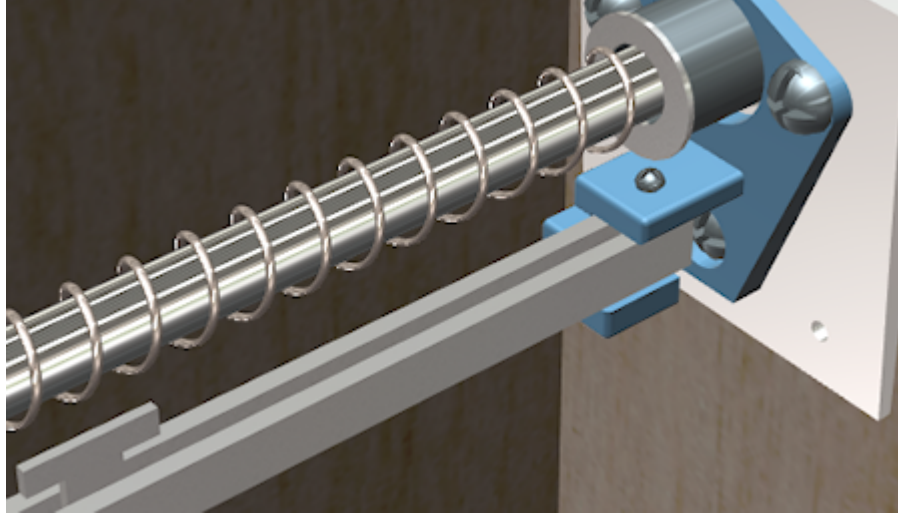
Now slide the bracket-and-pot assembly onto the end of the plunger, and all the way back until the bracket is sitting flush against the plunger mounting plate.



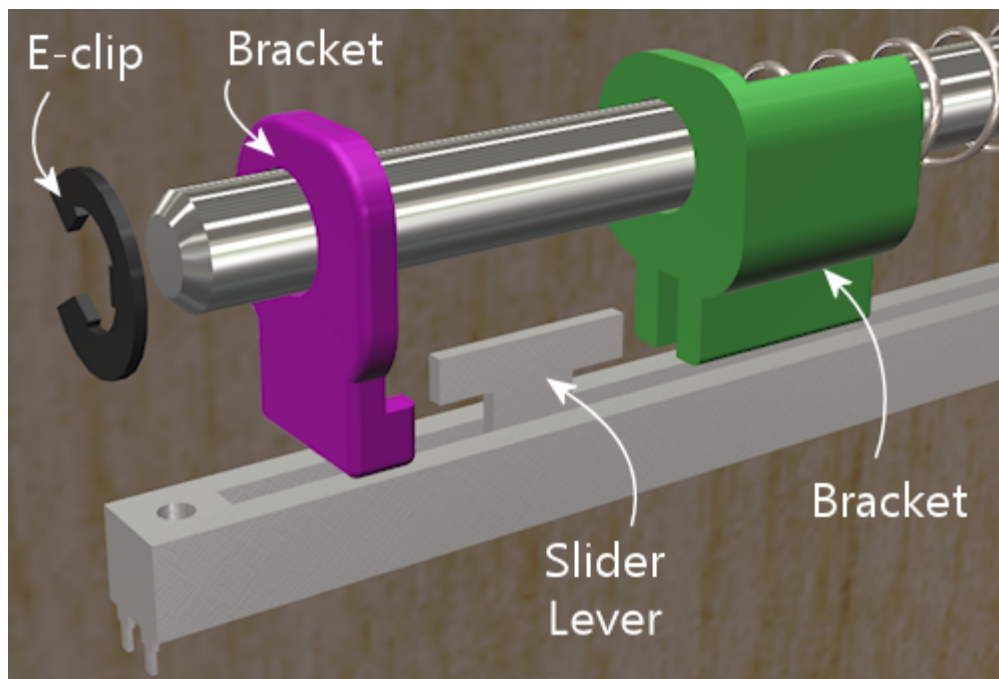


Reinstall the two top mounting screws that we removed from the plunger assembly at the start of the procedure. Fit them through the holes in the front plastic bracket. Don't over-tighten, as that could crack the plastic piece.

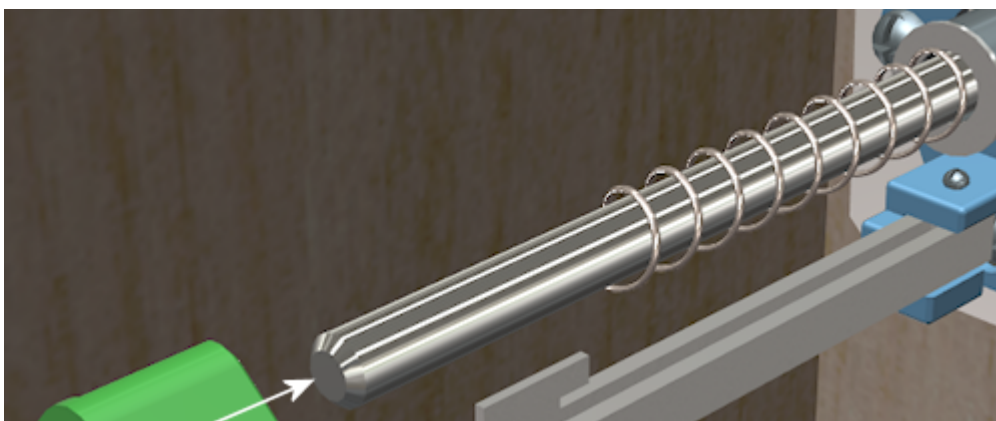


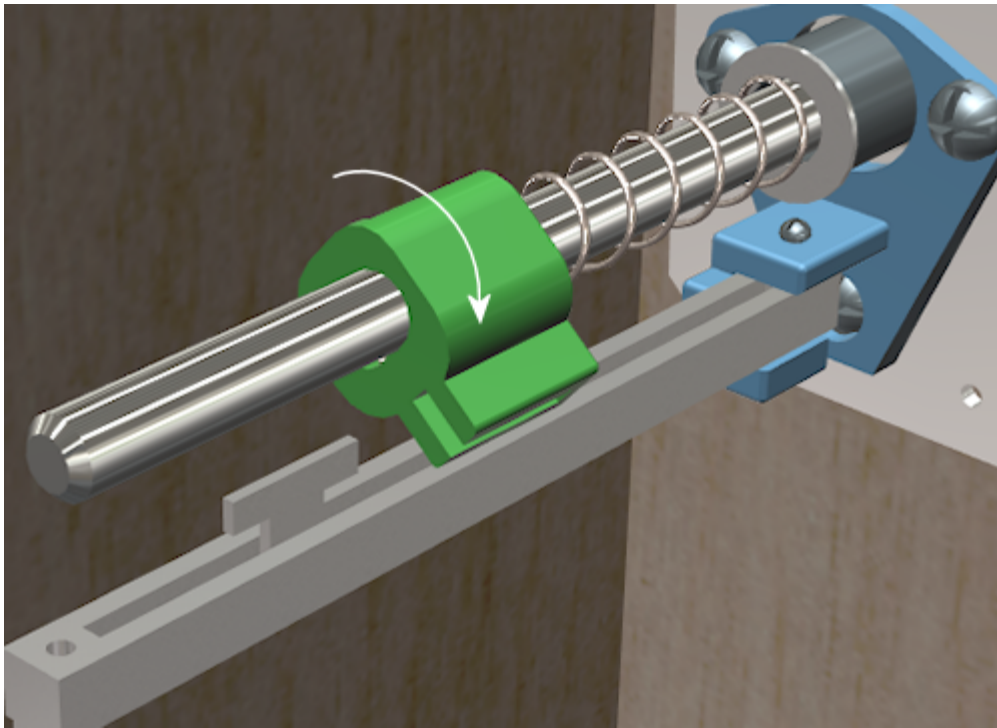


Now we're going to install the plastic bracket that connects the plunger to the slider lever on the pot. This connector bracket is actually two separate pieces. They work together to form a clamp that grips the pot lever from the front and back. The brackets fit over the plunger rod, between the spring and the E-clip. The tension of the spring holds everything in place.

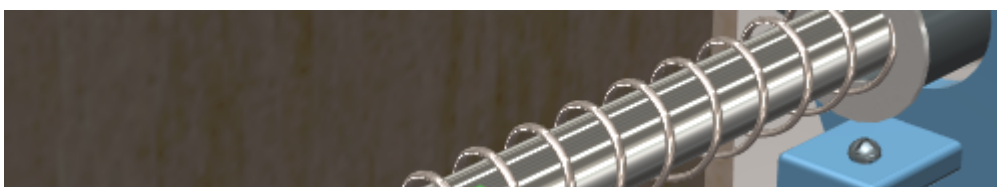
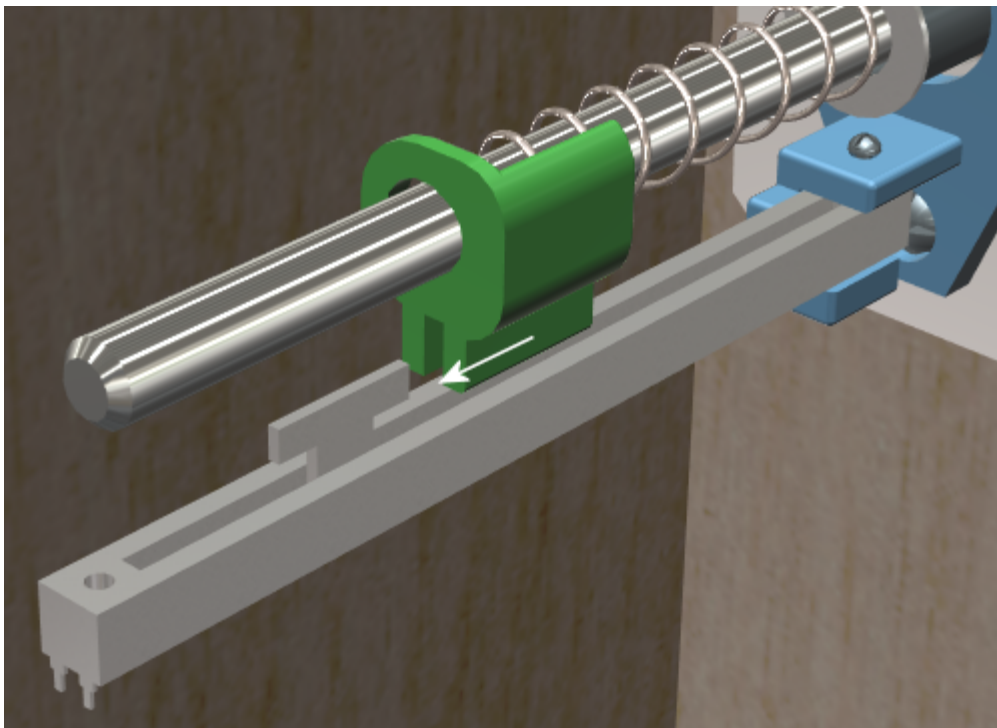


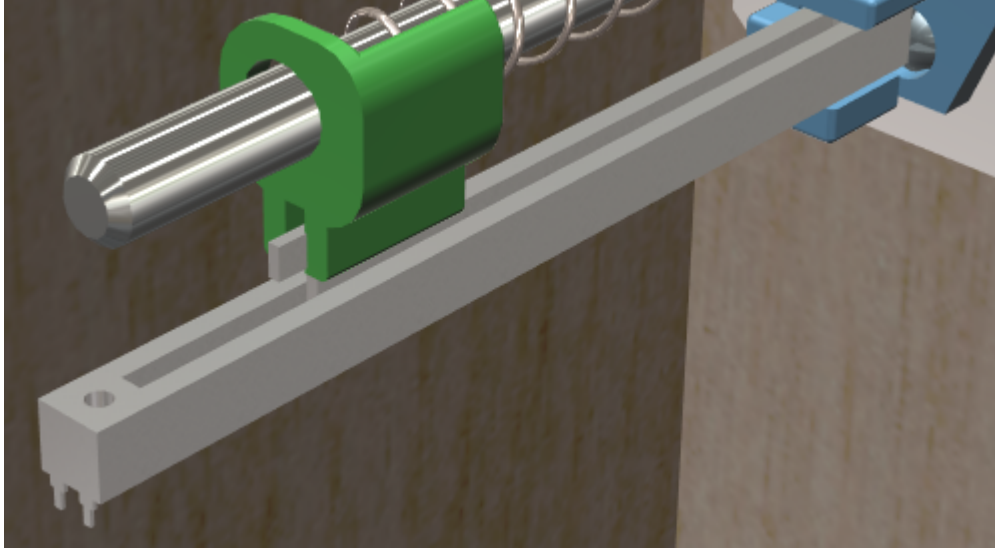
Install the larger bracket first, by slipping it onto the plunger rod as shown below and maneuvering it behind the slider lever. You'll have to compress the spring a bit as you put it in place.



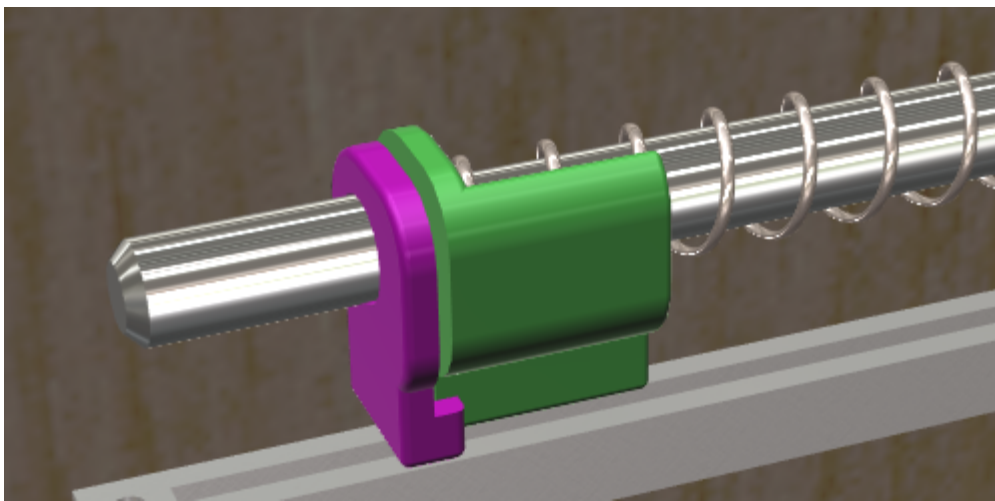
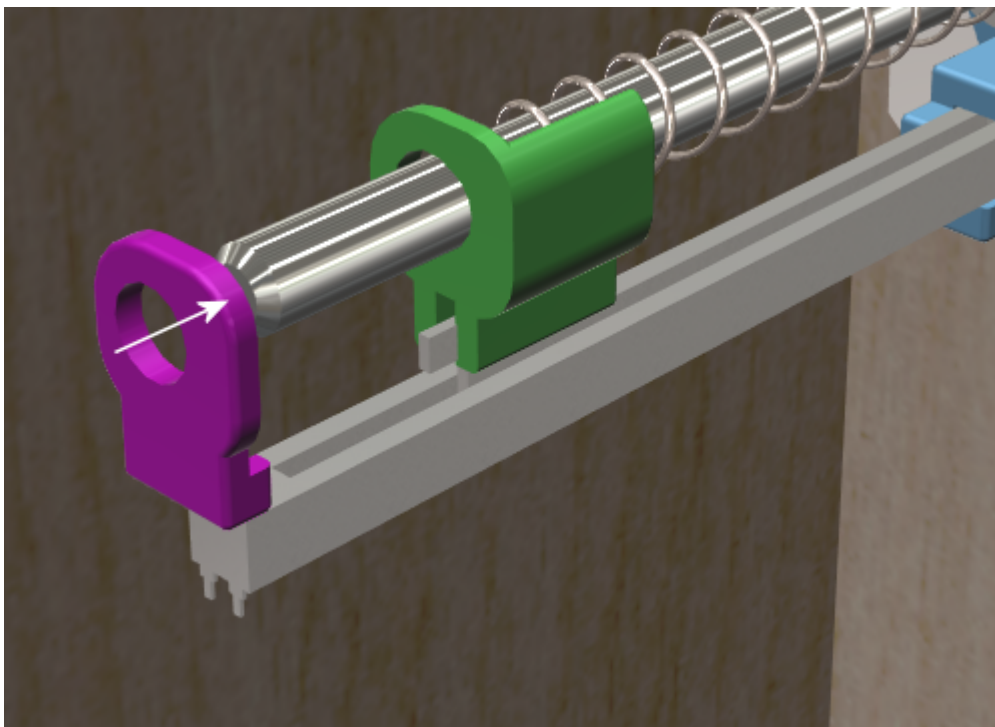


Slide the bracket over the potentiometer lever.



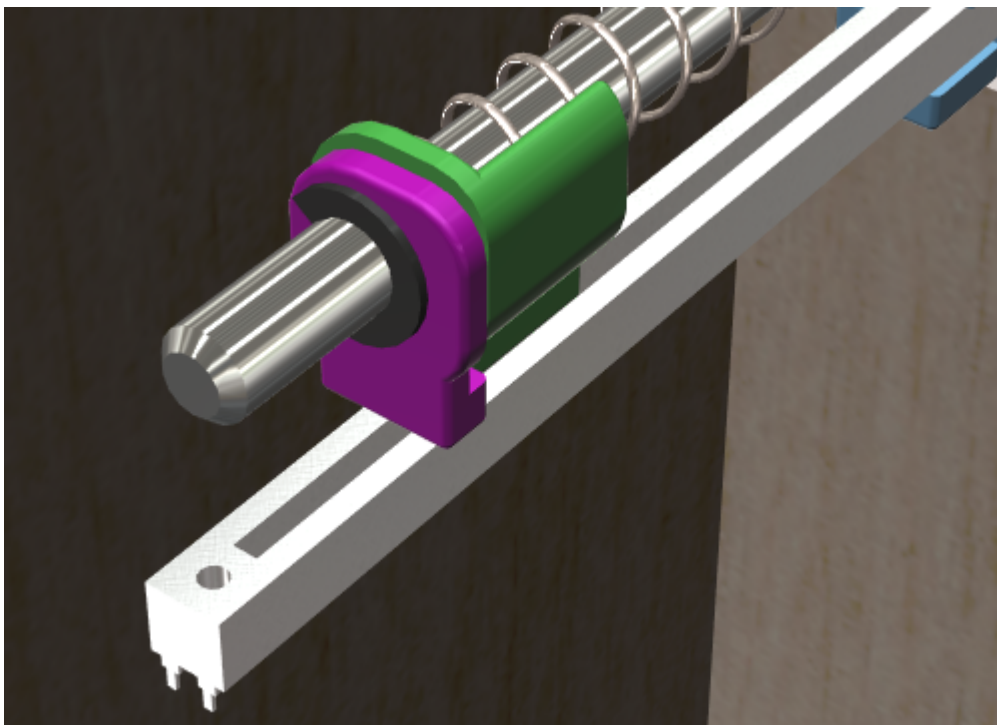
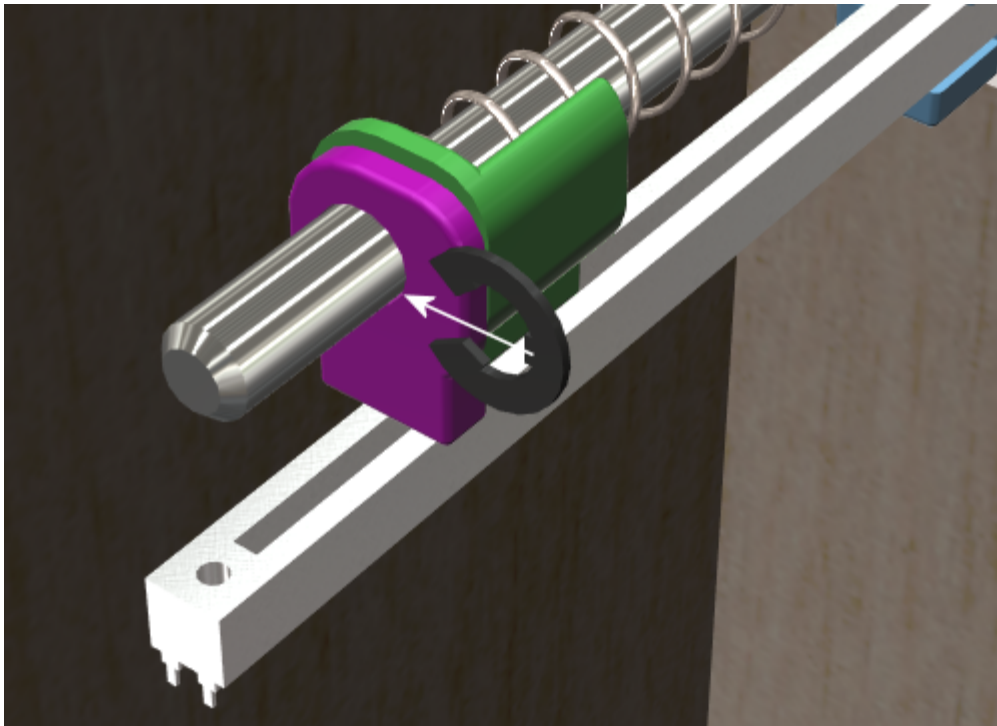


Slip the smaller bracket onto the plunger rod, and slide it back until it meets the larger bracket. The two should clamp over the pot slider lever. There'll be a little space between the two brackets when you're done, because the slider lever on the pot is a little wider than the spacing between the brackets. That's intentional, so that the brackets clamp down on the slider to make a solid grip with no play.



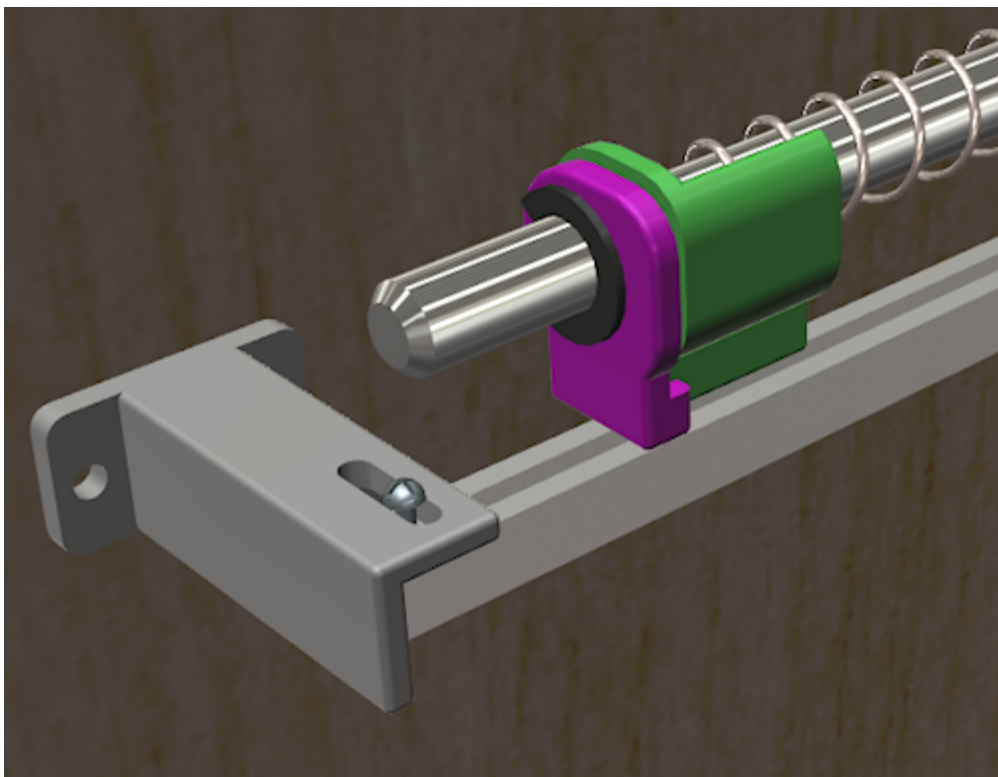
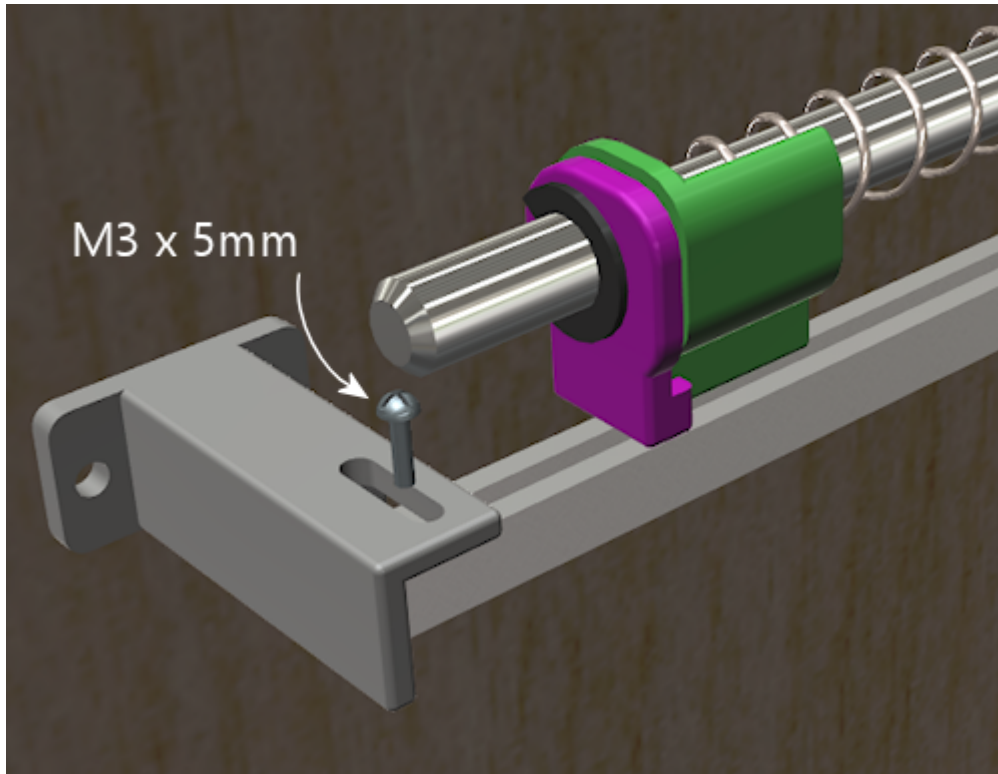


Push the brackets back against the spring far enough to make room for the E-clip to fit into the matching slot on the plunger rod, then re-install the E-clip. The E-clip just snaps into place with a little sideways force. I find it easiest to do this with needle-nose pliers.

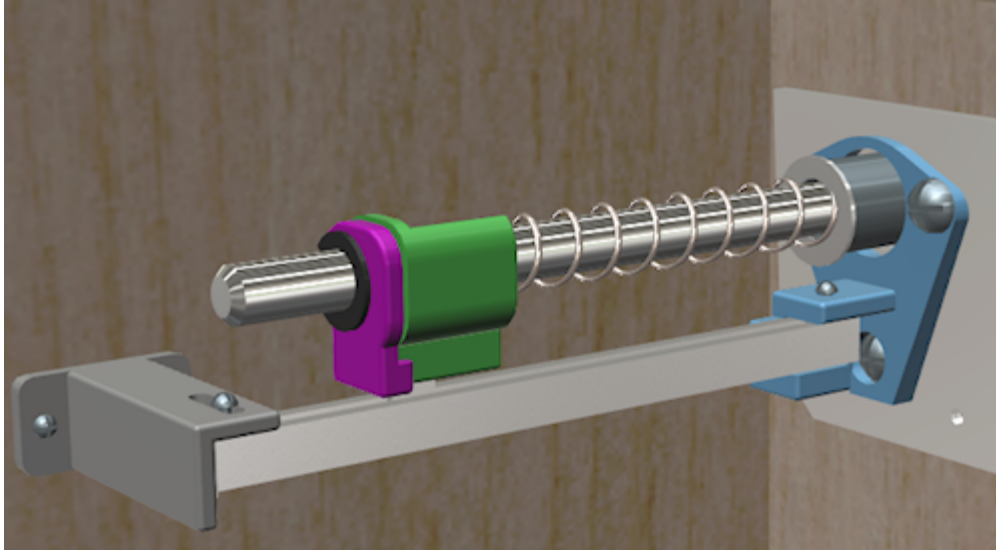


The last step is to install the bracket at the far end of the potentiometer. This bracket

attaches between the pot and the side wall of the cabinet. Fasten the bracket to the potentiometer with an M3 machine screw. Keep it loose for now so that we can fine-tune the positioning before tightening. (As before, the screw length should be from 5mm to 8mm long; choose the length that fits best with your parts.)



Before attaching the bracket to the wall, slide the plunger back and forth a few times to get an idea of how it affects the vertical positioning of the front of the pot. You'll want to pick a position for the bracket that allows the pot to move freely across its whole range, to avoid friction or stress on the mechanism. Once you've found the right position, screw the bracket into the side wall with a couple of #4 to #6 wood screws. (Sheet metal screws work equally well.)



Lemming77's design (not for ALPS)

Here's another 3D-printer plan (in Sketchup format) for a mounting for a different potentiometer, created by lemming77 on vpforums. **These plans are not the ALPS device from our parts list** - they're for some generic OEM part found a long time ago on Aliexpress, which is probably no longer available. You might be able to use these plans as a starting point, but be aware that **you'll have to modify them to fit your device's dimensions**.

- Mounting bracket
- Shooter rod connector



Lemming77's (vpforums) design for an older OEM potentiometer from Aliexpress, printed from the Sketchup plans linked above.

Software setup

If you haven't already set up your KL25Z with the Pinscape firmware, do that first. See Chapter 89, KL25Z Software Setup.


Start the Pinscape Config Tool. Click the Settings button for your device. Scroll down to the Plunger section. In the Sensor Type drop list, select Potentiometer.

If you're using the expansion boards, the pins should be configured automatically. If you're using the standalone KL25Z, set the "Wiper" field to the GPIO pin that you connected to the "variable" terminal on the potentiometer.

Save the new settings by clicking "Program KL25Z" at the bottom of the window.

You should now test and calibrate the plunger. Return to the home screen in the Config tool and click the Plunger icon for the unit with the sensor attached. This will let you look at the raw input from the potentiometer. When you move the plunger, the green bar in the setup window should move along with it.

If the green bar is responding, click the Calibrate button in the plunger viewer window. This will begin the calibration process. Follow the on-screen instructions. The plunger should be ready to use when the calibration process finishes.

If you're not seeing any response on the green bar, you might want to go back to the Settings window and make sure that the settings are right. Confirm that you've selected the Potentiometer sensor type, and double-check that the GPIO port you selected is the one you physically wired on the KL25Z. Also make sure that the pin isn't marked with a warning icon (). If it is, click the icon to see what's wrong. In most cases, the problem is that the same pin is assigned to multiple functions. If so, go to the other place where the pin is assigned, and clear that entry by setting it to "Not Connected".

If the software setup looks okay, check the physical wiring. Inspect each wire between the KL25Z and the potentiometer. Make each wire is connected to the right pin at both ends (on the KL25Z and the potentiometer). Check that the GPIO you assigned in the software matches the physical pin on the KL25Z you've wired.

Backwards operation

If the on-screen plunger moves backwards from the real plunger, you can fix it in the software without reinstalling the sensor. Open the Pinscape Config Tool. In the row for the controller, click the Plunger icon. Check the box for "Reverse orientation". (If it's already checked, un-check it.) This tells the software to reverse the readings from the sensor, so that it acts like it was installed in the opposite orientation.

103. Plunger Setup (VCNL4010 Proximity Sensor)

Vishay's VCNL4010 is an IR proximity sensor. It works by projecting pulses of infrared (IR) light in the direction of the target, and measuring the amount of light reflected back. If enough light is reflected back, the chip detects that an object is present within the target range. The brightness of the reflected light can also be used to infer the distance to the target, at least on a relative scale, since an object reflects more light when it's closer to the sensor than when it's farther away. It's this ability to measure distance that makes this interesting as a pin cab plunger sensor: we can use it to track the plunger position by pointing the sensor at the end of the plunger rod and taking distance readings.

I added support for this sensor mainly because several people who owned the VirtuaPin plunger kit requested it, so that they could migrate their kit to the Pinscape firmware while continuing to use the plunger sensor that came with their VirtuaPin kit. The VirtuaPin controller is based on the same retail KL25Z board that Pinscape runs on, so you can re-flash the board with the Pinscape firmware if you really want to, but until recently, doing so required replacing the VirtuaPin plunger sensor with a different sensor that the older Pinscape software supported, such as a potentiometer or a quadrature sensor. Now that the Pinscape firmware also supports the VCNL4010, you can migrate without changing any of the hardware.

What if you don't own the VirtuaPin plunger kit - should you still consider this sensor? My answer is a qualified "yes": it has some really good features, but it also has some drawbacks. If you've read the chapter on the somewhat similar VL6180X sensor, or if you've seen some of my posts on the forums related to IR sensors, you probably know that I'm not too keen on the various IR sensors in general as pin cab plunger sensors. Most of them just aren't very good at measuring distance. But this chip actually produces pretty decent results. It's not quite as good as a potentiometer or the quadrature or imaging sensor options (which are described in other chapters in this guide), but it's good enough to be quite usable. In my testing, it was able to resolve the plunger position finely enough over most of its travel range to produce smooth and responsive animation. This sensor's big weakness is accuracy. Its accuracy is good when the plunger is close to the sensor, but it drops when the plunger is farther away, so you might see somewhat chunky animation at the far end of the plunger travel, when the plunger is pulled back most of the way. In addition, the sensor's response curve (measured brightness vs. distance to the target) isn't perfectly uniform, so the distance calculations are non-linear. The Pinscape software tries to compensate for the non-linearity with an adjustment formula, but the actual response curve is a little bit erratic, so I haven't been able to make the response perfectly linear even after compensation. You might or might not notice this, depending on how closely you look. Even with these limitations, this sensor is still a good choice, given its other virtues: it's really easy to set up (maybe even easier than a potentiometer), it's inexpensive (under \$10), and it's non-contact (there's no wear from moving parts). Those features are nice enough that you might be willing to compromise a bit on accuracy.

Firmware and Config Tool versions

Support for the VCNL4010 is new as of June 2021. Make sure you're using versions of the Pinscape firmware and Config Tool dated 2021-06-01 or later (or, better still, just grab the latest release).

Electronics

You can buy a fully assembled VCNL4010 circuit board from Adafruit. This is lucky for us, because the bare chips are challenging to solder by hand (they're really intended for machine soldering). The pre-fab boards make them easy to use.

Adafruit VCNL4010 breakout board (\$7.50 + shipping)

The Adafruit board has easy-to-use solder terminals, and it can be wired directly to the KL25Z. No other hardware is required (besides wire).

If you have the VirtuaPin plunger kit, it happens to include the Adafruit breakout board, with the connections to the KL25Z already wired up. So you're already set!

Wiring the board

The Adafruit breakout board comes fully assembled, except for external wire connections. The only thing we have to do is attach the wires between the board and the Pinscape controller.

If you have the VirtuaPin kit that includes the Adafruit board, it's already wired, so you can skip straight ahead to the software setup.

The board comes with a pin header strip that you can optionally solder to the pads on the board. You can use these if you'd like, but I'd actually recommend soldering wires directly to the pin pads on the board instead of using the pin headers. Using the pins creates a bit of extra work, since you'll have to build an extra connector to plug into the pins. It also creates more opportunities for troubleshooting loose connections. So I'd skip the pins and solder wires directly to the board terminals, since you really shouldn't ever need to disconnect the wires from the sensor board. I do recommend using pluggable connectors at the KL25Z end, though, so that you'll still be able to easily unplug the whole assembly if you ever need to remove it from the cabinet.

I recommend 24 AWG stranded wire for the connections, but the exact gauge isn't important, as these wires carry very low power.

You'll need four wires, about three feet long each. Before cutting the wires, check the length you'll need by measuring the distance the wire will have to traverse to reach from the sensor location to the Pinscape unit (the KL25Z or main expansion board). Take into account any extra length you need to route it around your PC motherboard or other obstructions.

Strip about 1/4" of insulation from each end of each wire.

Solder one wire to each of the following terminals on the board:

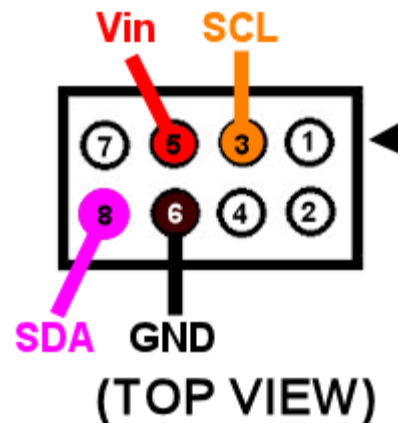
- Vin
- GND
- SDA
- SCL

Leave the other terminals (3v0 and INT) unwired. Don't connect anything to those terminals.

Expansion board wiring

For the expansion board connector, build a 4x2 crimp pin housing (housing, pins). First crimp a pin to the end of each wire (see Chapter 82, Crimp Pins). Insert the pins in the housing, following the diagram below for the pin placement.

The diagram below shows the layout of the 2x4 connector housing that you'll plug in to the expansion boards. The labels Vin, SCL, SDA, and GND correspond to the terminal markings on the Adafruit board. Connect the wire for each pin in the housing to the corresponding terminal on the Adafruit board, matching up the labels. Simply leave the unmarked pins empty in the connector housing.



It would be a good idea to put a mark of some kind on the housing in the corner next to pin 1 - the pin marked with the arrow on the diagram above. That will make it easier to remember which side aligns with the "pin 1" arrow marker on the circuit board when you plug in the connector.

Standalone KL25Z wiring

For the standalone KL25Z, I recommend using the Chapter 100, plunger sensor breakout board. Then you can just plug it into the breakout board.

- Build the expansion board connect as described above
- Build the ribbon cable connector exactly as described above, as though you were using the expansion boards
- Follow the instructions in Chapter 100, Plunger Sensor Breakout Board to build the board
- Connect the following wires between the plunger breakout board and the KL25Z:
 - Breakout board **3.3V** to KL25Z P3V3 (pin 8 on J9)
 - Breakout board **GND** to KL25Z GND (pin 12 or 14 on J9)
 - Breakout board **E20** to KL25Z PTE20 (pin 1 on JP10)
 - Breakout board **D0** to KL25Z PTE21 (pin 3 on JP10) (Yes, the label on the breakout board is different in this case)

The plunger sensor breakout board makes the connection a little tidier, but that's all it does - if you prefer, you can just wire the Adafruit board directly to the KL25Z pins. If you want to go that route, see "Plug it in" below for the list of the required KL25Z GPIO pin connections.

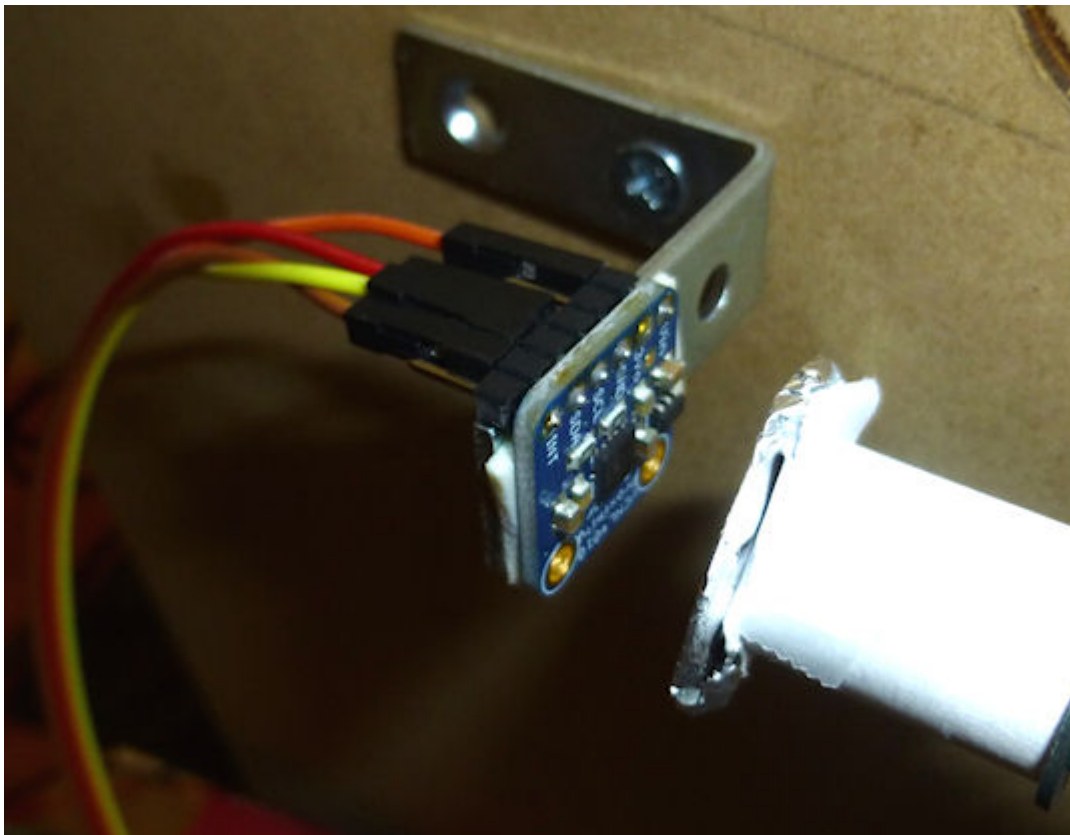
Physical installation

If you're using the plunger sensor from the VirtuaPin kit, you can simply continue using that whole apparatus, including its mounting bracket and wiring.

If you're building this all from scratch, you'll have to come up with your own way of mounting the sensor. I haven't come up with a reference design yet, but I can at

least describe the setup I used in my test rig:

- Materials:
 - 1½" angle bracket
 - #6 wood screws, ½"
 - double-sticky foam tape
- Mount the sensor to the angle bracket with the foam tape. Pick the location so that the sensor lines up with the plunger when the angle bracket is installed.
- Attach the angle bracket to the side wall of the cabinet with the wood screws. Position it so that the sensor is aimed directly at the plunger. Leave about 1.5 centimeters of clearance between the plunger and the sensor.
- The sensor must be far enough away from the plunger that there's no chance that the plunger will strike it when pulled back all the way and released. (You don't want all of that energy smacking into the sensor.) Test this by pressing the plunger all the way forward to make sure it doesn't hit. But it should be as close as possible after leaving enough clearance, since we're operating at the limits of the sensor's distance range as it is.



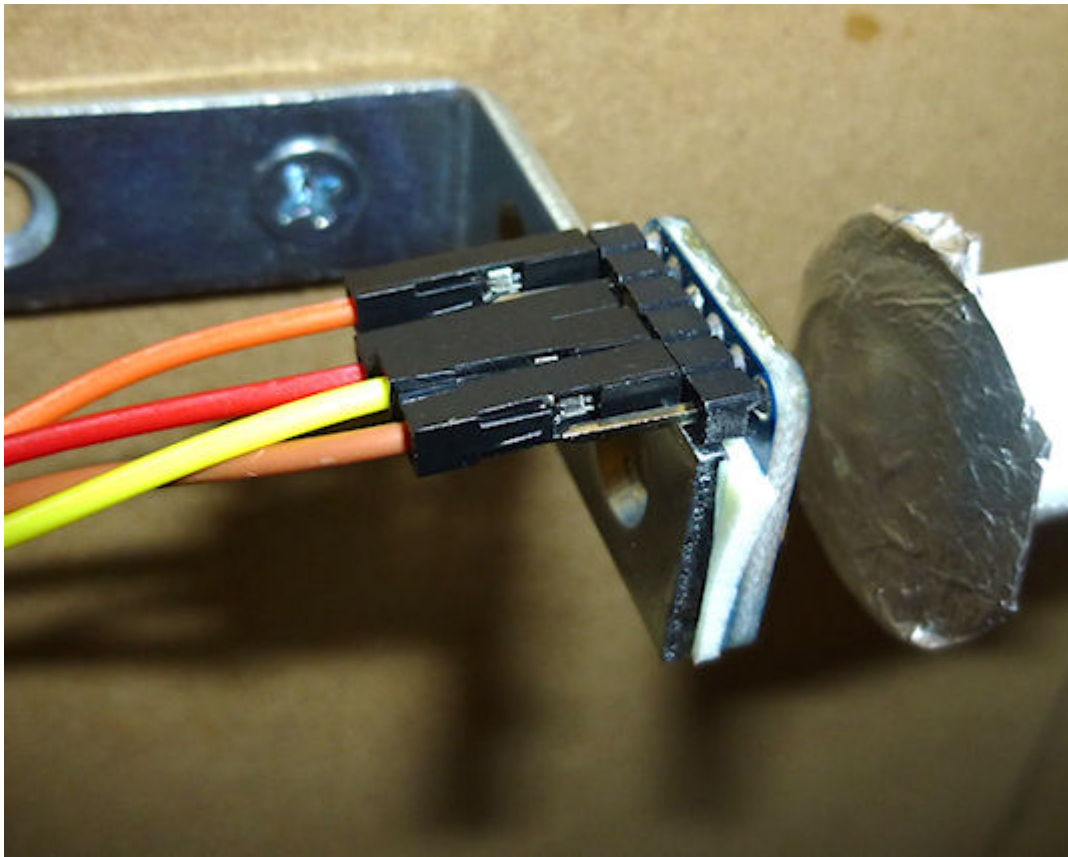
It would be nicer to come up with a 3D-printed plastic bracket that's more specifically designed to hold the circuit board, but I don't currently have a design for that to give you. If you're motivated to design and test a housing, and you want to share it, I'd be more than happy to include it in this guide if you send me a copy.

Reflector

You'll also need some kind of reflector attached to the end of the plunger. A standard white rubber plunger tip might be adequate, but you'll probably need something larger for good results.

Make the reflector as large as you can without hitting the flipper button switches or anything else installed nearby. A disk shape is ideal, since it'll provide a uniform

reflection even if you turn the plunger knob. (This would be another great place for a 3D-printed plastic part, but as with the mounting bracket, I don't currently have a design to offer, so I can only suggest improvising.)

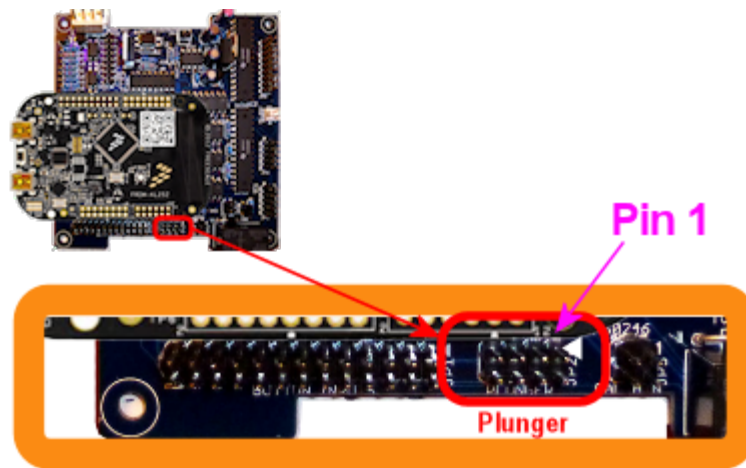


For test rig, I used a 1-inch disk, taped to the end of the plunger (as you can see in the photo above). I tried several materials to see what worked best. I had the best results with high-brightness white printer paper, and with kitchen aluminum foil, with the dull side facing the sensor. (The shiny side doesn't seem to work as well, probably because its reflected light isn't diffuse enough to produce a uniform reading at the sensor.) The foil performed very slightly better than the white paper in my test rig, but the difference was marginal enough that your results might vary, so I'd recommend experimenting with a few different materials to see what works best for you.

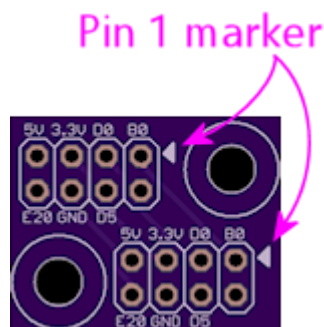
Don't rely on how bright a material looks to your eye to judge how good a reflector it will be. The sensor works in the infrared part of the spectrum, and some materials that reflect visible light well aren't as reflective in the IR spectrum (and vice versa). It's better to test a material rather than going by how it looks to the eye. You can use your live plunger setup as a test instrument, by looking at the brightness numbers in the Plunger Viewer window in the Pinscape Config Tool. That tells you exactly what the sensor is seeing. Pay particular attention to the brightness level with the plunger pulled **all the way back**. That's the most distant point from the sensor, which makes it the most challenging region for the sensor. It's useful to look at the brightness number for each material, but it's more important to look at how the brightness number varies with small plunger movements near the all-the-way-back end. In order to detect motion, the sensor has to be able to see changes in the brightness level that correlate with changes in the distance. So when you move the plunger by a few millimeters while it's nearly all the way back, you should see the brightness number go down as the plunger moves away from the sensor, and up as the plunger moves closer to the sensor. If you can see that pattern reliably with the sensor near the all-the-way-back end of its travel, your reflector should work well.

Plug it in

Expansion boards: Once you've built the connector as shown above, simply plug it into the plunger connector on the main expansion board. Make sure the plug orientation is correct by matching pin 1 in the housing (see the diagram) with the pin 1 triangle printed on the expansion board.



Standalone KL25Z: If you're using the plunger sensor breakout board (recommended), build the expansion board connector as described above, and just plug it in to the pin header on the breakout board. Be sure pin 1 on the plug (see the diagram) to pin 1 on the board, which is marked with a little white triangle printed next to the header.

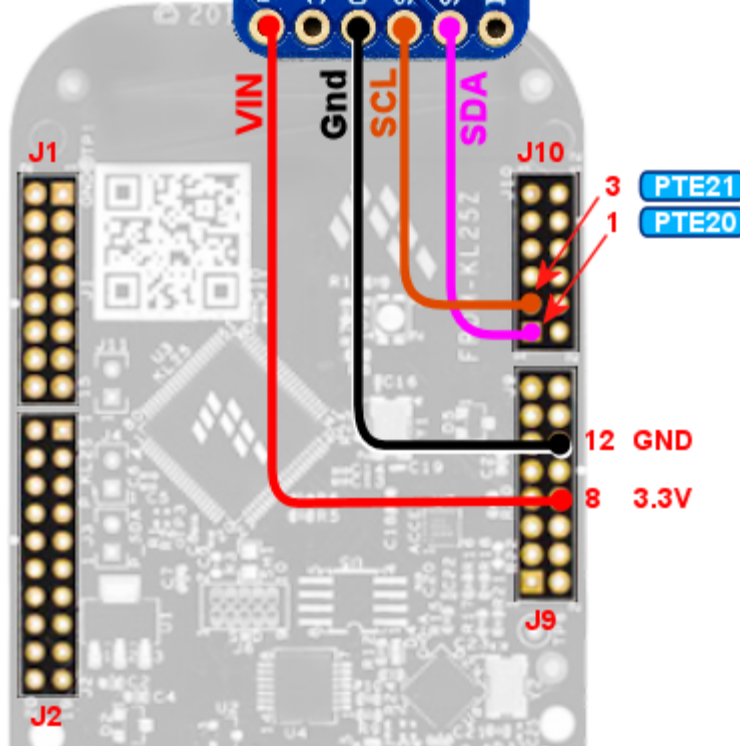


If you prefer to use your own ad hoc wiring, connect the wires between the board and the KL25Z as shown below.

Adafruit Board Pin	KL25Z Pin
Vin	P3V3 (JP9-8)
GND	GND (JP9-10)
SDA	PTE20 (JP10-1)
SCL	PTE21 (JP10-3)

Leave the other pads on the VCNL4010 board (3vo, INT) unconnected.





Note that the GPIO ports listed above are only suggestions. If you're already using the listed ports for other functions, you can assign the sensor inputs to other ports using the Config Tool. Any free GPIO ports can be used with this sensor - it doesn't have any special requirements for particular ports. The power and ground wires aren't configurable, though, so connect those as shown.

Software setup

Once you have the sensor physically installed and plugged in, run the Pinscape Config Tool on your PC. Go to the Settings page. (If you have multiple Pinscape units installed, choose the Settings page for the unit that's plugged into your new plunger sensor.)

Go to the Plunger Sensor section. Select VCNL4010 in the "sensor type" popup.

(If the VCNL4010 option isn't available in the plunger sensor list, you probably have an older version of the Config Tool. Updating to the latest version should add the option.)

If you're using the expansion boards, the appropriate pin settings will be selected automatically.

If you're using the standalone KL25Z, set the pin assignments for the two I/O pins (SDA and SCL) to match the pins you connected on the KL25Z. The SDA and SCL pins should match the pins you wired to the like-named terminals on the sensor board.

If you have the VirtuaPin kit with the VCNL4010 board, it's already wired to the KL25Z, so all you have to do is set the pins to match the kit's wiring. The pin assignments for the kit are **SDA = PTC2, SCL = PTC1**.

Here's a quick summary of the pin assignments, depending on your configuration:

Configuration

SDA

SCL

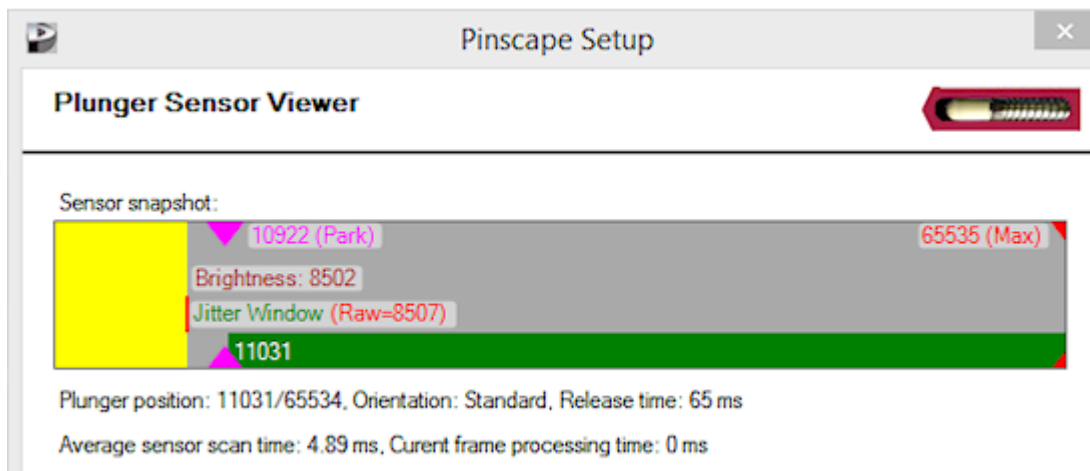
Expansion Boards	Automatic	Automatic
Standalone KL25Z, with plunger breakout board	PTE20	PTE21
Standalone KL25Z, with your own ad hoc wiring	As you wired it	As you wired it
VirtuaPin plunger kit	PTC2	PTC1

There's also a box where you can enter the "IR LED current" setting. This adjusts the brightness of the IR emitter on the chip that's used to produce the light beam for the distance measurements. You can experiment with this to see what produces the best results in your setup, but I recommend starting with the maximum value of **200 mA**. The VCNL4010 can only barely detect objects as far away as the plunger travels, so it needs the brightest light source you can supply to be able to track the plunger.

Note that the 200 mA power setting doesn't mean that the sensor will actually draw 200 mA from the power supply. That power level might look alarming, but you really don't have to worry about it. The sensor only generates brief pulses of light, so it uses much less power overall than this setting suggests. According to the data sheet, the sensor only draws 4 mA overall when set to its highest power setting (with the IR LED set to the maximum 200 mA, and the sampling speed set to the fastest rate). 4 mA is a negligible amount of power that won't affect the KL25Z or your USB ports. This sensor is designed for use in battery-powered devices, so they made it quite efficient on power.

Save the new settings by clicking "Program KL25Z" at the bottom of the window.

You should now test and calibrate the plunger. Return to the home screen in the Config Tool and click the Plunger icon for the unit with the sensor attached. This will let you look at the raw sensor input. Here's what the plunger viewer visualization looks like for this sensor:




The yellow bar shows the "brightness" of the IR light signal that the sensor is receiving. Remember that this sensor doesn't actually measure distance - all it can measure directly is the IR light level it's receiving on its photo sensor. The yellow bar lets you see exactly what the sensor is reading on its IR receiver. The brightness level is in abstract units that range from 0 (complete darkness) to 65535 (full saturation, like an overexposed photo). The 0-65535 scale is what the sensor actually reports on its digital interface to the KL25Z, so this shows you the unfiltered data directly from the sensor.

The green bar superimposed over the yellow bar shows the plunger position that the software is computing from the brightness reading. Remember that this isn't something that the sensor measures - the sensor only gives us that brightness reading. The plunger position has to be inferred from the brightness. This calculation is only meaningful after calibration, so the green bar will usually act erratically (or it might even be missing entirely) when you first hook things up. Don't worry about the green bar right now - let's just make sure the sensor is working first, by checking that the raw brightness reading (the yellow bar) is behaving properly.

To check that the sensor is working, try moving the plunger, and watch the yellow brightness bar. It should change when you move the plunger. The brightness should decrease (lower numbers) as you pull the plunger back, since the amount of light reflected from an object decreases as the object moves farther away. You might immediately notice that the yellow bar's motion isn't "linear" - the bar doesn't track changes in the plunger position at the same rate across the whole travel range. That's exactly as it should be! The yellow bar is showing you what the sensor sees at the physical level, and at the physical level, light intensity is related to distance through an inverse power law. As a result, the yellow bar moves faster when the plunger is close to the sensor, and slower when the plunger is further away.

You might also notice that the yellow bar doesn't cover the entire numeric range from 0 to 65535 as you move the plunger. That's also to be expected. You might see it get close (or all the way) to the maximum 65535 when you push the plunger right up against the sensor, but it probably won't fall to anywhere near zero when you pull it all the way back. This is because the sensor always registers a certain baseline background light level, which comes partially from ambient light, and partially from the sensor's own internal electronics. This minimum level apparently varies quite a lot from one sensor to the next, according to the data sheet, so don't worry if the yellow bar bottoms out with a number in the thousands. It's also better if the numbers never quite reach the maximum, because that means you're getting the plunger so close to the sensor that it's getting "overexposed", where it can't distinguish distances. You might back the sensor away from the plunger by a couple of millimeters if the yellow bar hits 65535 before the plunger is absolutely all the way forward.

If the yellow bar isn't there at all, or it doesn't move when you move the plunger, the sensor either isn't working or isn't connected properly to the software. Go back to the settings page and double-check the GPIO pin assignments. **Make sure that none of the pins are marked with error icons** (). If you see any of those, click the icon to see details on the conflict. These conflicts **must** be resolved - a lot of people see the icons and think they're just just some kind of advisory that can be ignored (I guess we're all used to Windows software constantly pestering us with meaningless warnings and error messages), but in this case you really have to pay attention to them. The sensor simply won't work if there are any pin assignment conflicts. If there are no conflicts, trace each wire and make sure that it actually goes to the assigned pin on each end (KL25Z and sensor board). Check that each GPIO port assignment on the settings page matches up with the physical pin on the KL25Z and connects to the corresponding terminal on the sensor board. Try to look at it with a critical eye - I find that it's really easy to trick yourself into seeing what you want to see when doing this kind error checking.

If the yellow brightness bar behaves as expected, click the Calibrate button in the plunger viewer window to begin the calibration process. Follow the on-screen instructions.

Make sure that everything's set up the way it would be during normal operation when you run the calibration. For example, if you opened up your cabinet or

removed the TV to install the plunger hardware, you should put everything back together for calibration. The VCNL4010's brightness readings can be affected by ambient light and reflections from nearby objects, so the calibration results will only be accurate when the environmental conditions are the same as when you ran the calibration. You might need to repeat the calibration process if you make any substantial changes to your cab in the future that might affect ambient light inside the cabinet or similar factors. (Things like the ambient light level in the room *outside* the cabinet shouldn't matter, since the cab should block all of that.)

Jitter filtering

The VCNL4010 has some inherent electronic "noise", or randomness, in its brightness readings. If you watch the brightness number on the yellow bar in the plunger setup dialog, you'll see that it jumps around over a small range when the plunger is sitting perfectly still. This is due to random variations in the brightness readings on the sensor. A certain amount of random noise is a natural part of any analog measurement process.

The Pinscape software has a "jitter" filter to help reduce the visible effects of the noise. The jitter filter lets you set a range of expected random fluctuations; the software will ignore changes within that range. As long as the random noise from the sensor stays within the range, the device ignores the fluctuations and reports a stable plunger position that remains stationary when the real plunger isn't moving, and tracks the real plunger's position smoothly when it's moving.

To enable the jitter filter, run the Pinscape Config Tool and go to the Plunger Viewer window. There's a setting in this window for the jitter filter. To adjust it, start with the filter at zero, and gradually increase it until the green bar showing the filtered reading stops jumping around. Use the smallest value that gives you stable readings. If set the filter value too high, the animation will start to get chunky.

For my test setup, a jitter window of 10 to 20 works nicely, but you should experiment. The optimal setting will vary by system, since so many different physical factors can affect the noise level in the IR signal.

Theory of operation

The VCNL4010 is what's called a "proximity" sensor. It's designed to answer the yes/no question, "is there an object near (in proximity to) the sensor?".

That's not exactly the question that we want to ask of a plunger sensor. Our question is, rather, "how far away is that object?", and we'd like to take our answer in millimeters, or some other unit of distance. Technically, a sensor that's designed to answer the question "how far?" is a *distance* sensor. Using a proximity sensor like the VCNL4010 to answer "how far?" is a little bit of an abuse of the chip, since we're asking more of it than its designers intended. But even so, it's still a question we can meaningfully ask with this particular chip, because the chip gives us more information than just a yes/no answer.

What the sensor actually gives us, in lieu of directly answering "is an object there?", is something a little more quantitative, which we're meant to interpret as an answer to that question. The quantity the sensor measures is the intensity of the reflected light from the target object - the reflection of the IR beam that the sensor shines on the target. We're supposed to compare this to a threshold brightness level, which we have to determine by calibrating the sensor (by placing a test object at a known distance from the sensor, and taking a reading). After we've calibrated with a test object, we can start taking proximity readings. If a brightness measurement is above

the calibrated threshold, we're supposed to take that to mean that an object is nearby; if the brightness is lower than the threshold, we take it to mean that nothing's there. Technically, that's all the information this sensor is supposed to give us; we're not supposed to wonder exactly how far away the object is when an object is detected. But we could "abuse" the brightness reading by treating it as a continuous quantity, instead of just comparing it to a single threshold, using the physics principle that the amount of reflected light should be inversely related to the distance to the reflecting object. The reflected brightness is essentially a proxy for the distance.

That might seem perfectly reasonable, so why do I call this an "abuse"? Simply because it's not what the chip is designed for. The brightness might be usable as a proxy for distance, but it's not necessarily a *good* proxy. A good proxy would have a well-defined relationship to distance, so that we can precisely calculate the distance corresponding to any given sensor reading. This chip's designers weren't at all trying to achieve that, and the chip's data sheet is silent on exactly what the relationship to distance might be, other than providing some rough plots of test data for a simple reference setup.

The difficulty with figuring out the precise relationship between brightness and distance (and why the manufacturer doesn't attempt to specify the relationship in the data sheet) is that it depends upon many different factors: the size and shape and reflectivity of the target object, the sensitivity of the sensor, the amount of ambient light, stray reflections, and so on. For a pin cab plunger, we can at least try to arrange things so that most of those other factors are constant, so that the only variable is the distance. But even holding everything except distance constant, we still can't know the precise mathematical relationship between distance and brightness, so we have to figure that out empirically. The best we can do is observe the relationship through calibration, and then interpolate points in between the calibrated reference points. In the abstract, that might or might not work, depending on how uniform and repeatable the relationship between brightness and distance actually is in practice for the sensor.

Fortunately, for this sensor, it turns out that the relationship is uniform enough and precise enough that we can get pretty good results with it.

In my testing, the best fit to the data I gathered turned out to be simple inverse square relationship - the measured brightness is proportional to the inverse of the square of the distance. That's just what you'd expect from an idealized point light source, so it matches my intuition at that level, but it's actually somewhat different from what the Vishay data sheet suggests. As I mentioned, the data sheet doesn't specify a numerical relationship between distance and brightness, but it does at least provide a rough plot of some test data, and from that it appears that the relationship they observed is more like an inverse cube curve (it looks like about $1/r^{3.2}$ to me, but the data sheet plot is shown at such low resolution that it's hard to read it with any precision). Even though an inverse square law fits my intuition about an idealized point-light-source setup, my intuition also says that our reflector apparatus has some pretty significant geometry differences from a point source and that we therefore shouldn't trust such simple intuition. Even so, it seems to be what fits! But so far I've only tested this sensor in this one installation, so we'll have to see if the inverse square fit that the firmware is currently using works as well for other people. I can believe that the power-law relationship might vary a little bit based on the physical setup, and that we'll eventually need to make that configurable to get optimal results across different machines. But we'll see.

104. Plunger Setup (AEDR-8300 Encoder)

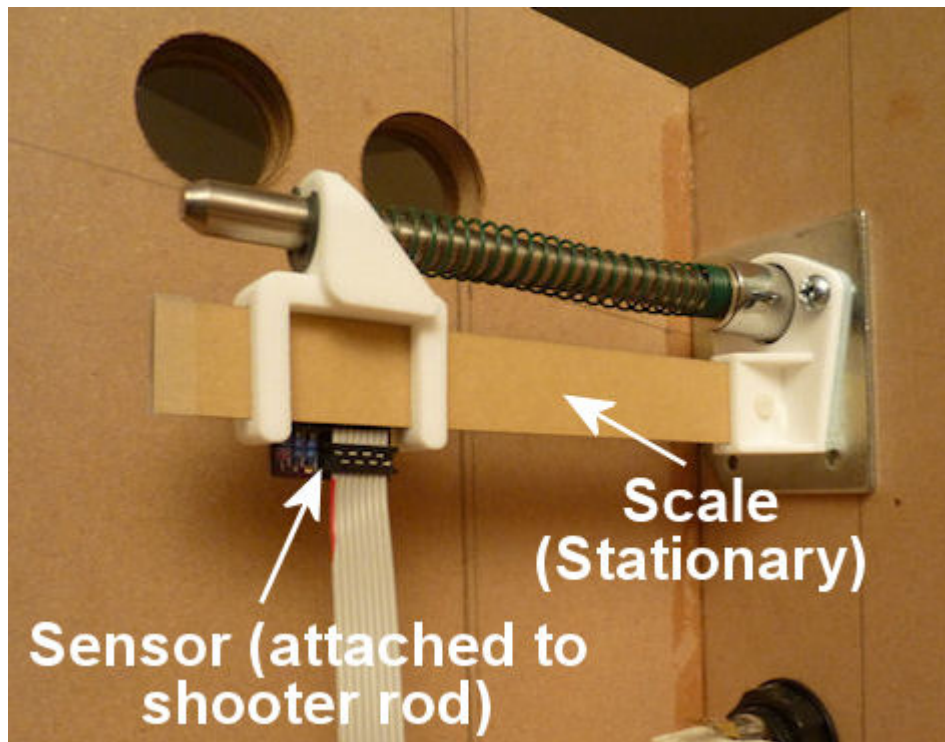
One of the newer sensor option for the Pinscape plunger is an "optical encoder" chip called the AEDR-8300.

An optical encoder works with a guide rail known as a "scale" to sense its position. The scale is marked with alternating black and white bars of equal width. As the sensor moves along the scale, it counts the bars it passes. Since the bars are all of the same width, the count tells us the position in units of the bar width. The AEDR-8300 uses very fine bars, about 6.6 thousandths of an inch wide, which lets it measure the position very precisely.



Pinball plungers are obviously well suited for this basic idea, since they're naturally constrained to move in one dimension. The position along the scale tells us everything we need to know about the plunger position.

This chapter explains how to set up this type of plunger sensor.



3D printing guidelines

- The STL files linked here use **millimeter (mm)** units
- Nylon materials are recommended (PA12 or PA11)
- MJF (multi-jet fusion) process is recommended

I don't recommend using a home 3D printer for these parts. Consumer-grade printers mostly use PLA or ABS, which aren't good for functional parts like these since they tend to disintegrate rapidly when exposed to friction. I recommend using a commercial 3D-printing service (All3DP.com, Shapeways.com, or 3DHubs.com), and choosing a **nylon** material (PA12 or PA11). If your vendor offers the newer MJF (multi-jet fusion) process, I'd consider that - it seems to produce extremely tough parts that should hold up well.

Parts list

This plunger project requires some specialized parts, all of which you can make yourself from the plans below. Just a few years ago, a lot of this would have been all but impossible for a hobbyist, but it's actually pretty easy now thanks to 3D printing and other technologies. In fact, the project is a nice tour of the many modern options for "personal manufacturing":

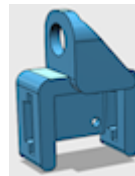
- Two of the parts can be 3D-printed
- One part can be made from laser-cut plastic
- One piece can be printed on a regular inkjet or laser printer
- One part is a custom circuit board

Fortunately, you don't need a factory at home for any of this, since there are companies that will do one-off 3D printing, laser cutting, and circuit board manufacturing at reasonable prices. See Chapter 4, Resources for my recommended vendors.

There are economies of scale in ordering some of these parts in batches, so you might want to consider finding a few people on the forums and place a group order. I've organized group orders in the past myself for these sensors, so feel free to contact me to see if I have any parts currently available.

If you want to order everything yourself, here are the pieces you'll need.

Sensor bracket. This plastic piece fastens the sensor to the plunger. It simply fits over the end of the plunger rod between the e-clip and the spring, so it's held in place by the spring.



This can be 3D-printed using these plans: `sensorBracket.stl` (units are millimeters).

As noted above, I highly recommend having this part made by a commercial 3D-printing service (**not** a home printer) using a **nylon** material, preferably with the MJF process.

Guide bracket. This plastic piece serves as the anchor for the guide rail, which holds the optical bar pattern ("scale") that the sensor scans. This fits over the metal plate that anchors the screws holding the ball shooter housing in place.



This piece can also be 3D-printed, using these plans: `guideBracket.stl` (units are millimeters).

Guide rail. This is a piece of 3mm thick mirrored acrylic that serves as the guide rail that the sensor moves across. This fits into the guide bracket and is held in place with a bolt.



This piece should be laser-cut using **3mm mirrored acrylic**. The mirrored surface is needed to get enough reflected light for the sensor to work properly. Laser-cutting templates are linked below. These are suitable for upload to Ponoko.com, which offers the right kind of mirrored acrylic as an option. The "single rail" file includes only one copy of the rail, whereas the "multi-pack" fills Ponoko's entire small sheet with copies. The multi-pack makes the best use of materials, and gives you lots of extra copies to share with friends. On the other hand, you can use the single copy version if you want to fill in the unused space with some other design of your own. This could be interesting for making custom decorations for your cab - the mirrored acrylic is great for making custom lettering or logo cutouts.

Single rail: GuideRailCuttingTemplate-x1.svg
Multi-pack: GuideRailCuttingTemplate-x11.svg

Printed scale. The bar pattern that the sensor reads can be printed with an ordinary laser printed on transparency sheets. Use transparency sheets made for laser printers, such as 3M CG3300.

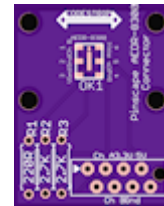


A PNG graphics file with the pattern is linked below. I recommend printing with a laser printer, not an inkjet, since inkjet ink isn't as opaque as laser toner. Use a high quality setting, since it's important to maximize contrast. If you have a color laser printer, print in monochrome mode with black toner if that option is available.

This image is at 600dpi, and should print at 5" wide. Your printer will probably scale it automatically if necessary to match its native resolution, but if you can't get it to print at the proper size or it comes out fuzzy, you should try manually rescaling it to your printer's native resolution with a graphics program. The exact size of the bars is important, so don't rescale the overall image to a different print size. The only adjustment you should make is to the pixel resolution.

Graphics for printing: scale.png

Circuit board. This small printed circuit board holds the sensor chip, plus a few resistors it needs, and pads for a ribbon cable to connect to the Pinscape expansion boards. This fits into the sensor bracket. You can order these from OSH Park simply by uploading the EAGLE .brd file.



EAGLE plans: pcb.zip

Electronics. You'll need the sensor IC itself, plus a few other electronic components that go into the circuit board. Everything is listed in the Chapter 91, Electronic Parts List in the AEDR-8300 plunger sensor section.

Please see the **warning on humidity** below. Don't open the AEDR-8300 plastic pouch until you're ready to solder it to the circuit board (even though it's tempting to open the package and check it out when you first get it).

Cable/wires. For the expansion boards, order the ribbon cable connectors listed in the AEDR-8300 parts list, plus an 8-conductor ribbon cable in whatever length you need to connect to the board. 3 feet should be more than enough. If you can't find an 8-conductor cable, you should be able to cut one with more conductors down to size. Most ribbon cables can be easily torn like a zipper along wire boundaries to reduce them to however many wires you need.

For the standalone KL25Z, it's most convenient to use individual wires (not a ribbon cable) to connect to the KL25Z. You can simply solder wires directly to the sensor board, and connect the other ends to the KL25Z via the 0.1" crimp pins listed in the KL25Z standalone section of the Chapter 91, Electronic Parts List.

Plunger. Of course, you'll also need a real pinball plunger. You don't have to fabricate anything for this; you can just buy a real one. The parts above are all designed around the Williams/Bally ball shooter assembly, Williams part no. B-12245. They haven't changed the design since at least the late 1980s, so if you have an older version, chances are it'll fit. I think the current Stern plungers are the same size as well, but I haven't confirmed that.



You should also buy the special mounting plate for the shooter assembly, Williams part #01-3535. It's only about \$2 and it makes it much easier to install.

Fasteners. The following fasteners are recommended:

- (Qty 1) M2x12mm or #4x½" machine screw and mating nut, preferably nylon, for attaching the acrylic guide to the bracket
- (Qty 4) M2x8mm or #2x¾" machine screws and mating nuts, preferably nylon, for attaching the circuit board to the sensor bracket
- (Qty 3) #10-32 x 5/8" machine screws, steel, for the ball shooter assembly housing

Group orders

For the smaller parts, particularly the acrylic rail and the circuit board, you can save money with a group order. These can be made in batches much more cheaply than as single copies. I might have a small supply on hand; if so, I'll be happy to send you parts from my batch at cost as long as you're in the US. Contact me on the forums to inquire (see Chapter 4, Resources).

The 3D-printed parts and electronics don't have any particular scales of economy, so I'd recommend ordering those individually. If you have your own 3D printer, you can print the 3D parts yourself.

Warning on humidity

When you order the AEDR-8300 from Mouser, they make a fairly big deal about its sensitivity to humidity. It'll come in a sealed plastic pouch with a big warning sticker about humidity exposure, and an indicator card sealed inside that changes color as it picks up moisture from the air. The card is there to verify that no moisture leaked into the packaging during shipping and storage, so check it when you first open the package. (The card will start changing color quickly after you open the package. Don't worry about that; it's there purely to assure you that the packaging was intact. If the card indicates that the packaging *wasn't* moisture-proof after all, use the "baking" procedure that we'll come to in a moment.)

The humidity warning is there because the plastic housing material used in this chip can absorb moisture from the air and trap little droplets of water in pores in the plastic. When you solder the chip, the heat will turn any trapped water into steam, and the sudden expansion can warp or crack the housing. It's like microwaving a sealed container. This can destroy the chip.

To avoid this danger, don't break the seal on the packaging until you're ready to install the chip. Read through the installation steps before you open the pouch, and make sure you have all of the necessary tools and supplies on hand before you start, so that you can complete the soldering job in one session once you start.

There's no need to panic, though. The guidelines for this chip say that it's okay to solder for up to 168 hours (7 days) after opening the package.

What happens if you go past the 168-hour deadline, or the moisture indicator card in the packaging shows a breach? The data sheet has a straightforward solution: "bake" the chip, placing it in an oven at 60° C (140° F) for 48 hours. That'll gently exorcise any trapped moisture and restore the chip to a happily desiccated state. That'll give you a fresh 168-hour window to complete soldering.

Moisture is only a worry during the soldering process. You don't have to worry about

humidity exposure once you've installed the chip on the board.

Assembling the circuit board

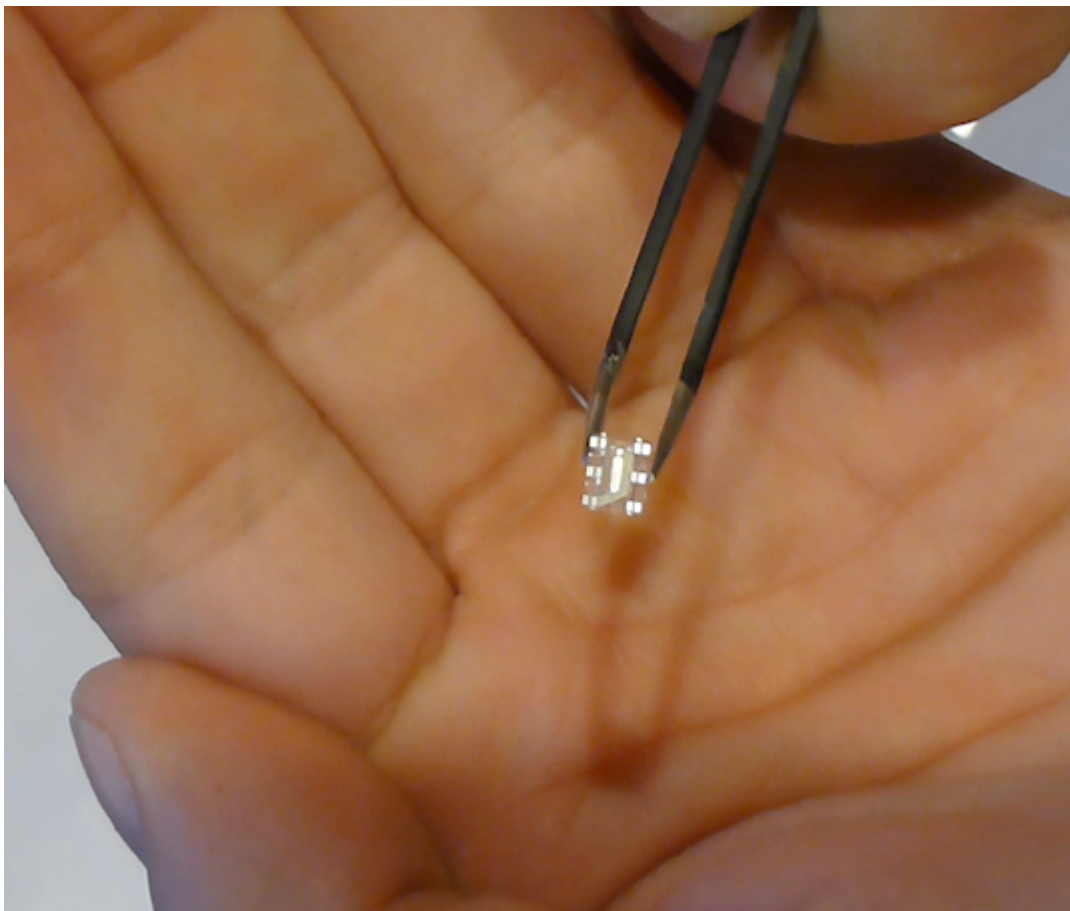
Please read the warning above on exposing the sensor to humidity before proceeding.

Take a look at the circuit board and check for any little tabs or spurs around the edges. Small boards like this are usually made as parts of larger panels, so there are sometimes a few rough edges left over. If you find any tabs sticking out, trim them with wire cutters or something similar. The board fits snugly into the 3D-printed plastic bracket, so spurs can prevent it from fitting properly.

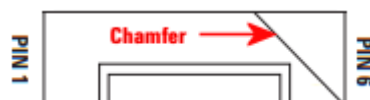
Installing the sensor chip

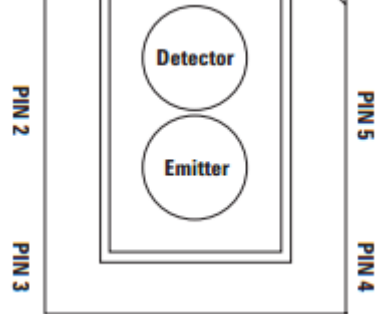
This is the only tricky step in assembling the board, and it should be done first.

The AEDR-8300 is a small surface-mount part. As you can see in the photo below, it's really tiny. I recommend having a magnifying glass and forceps at the ready while working with it.

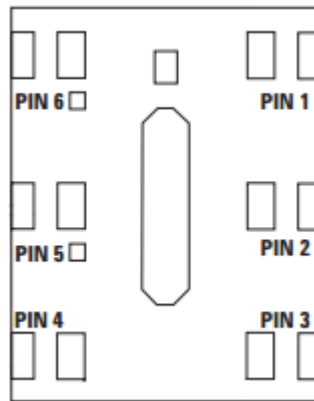


The first thing you need to do is figure out the chip's orientation. Start by identifying the front and back. The back is the side with the metal pads for soldering. The front is all clear plastic, with a couple of little circular bumps for the light source and sensor lens. Be careful: the whole package is made of transparent plastic, so you can see the metal pads from both sides. But if you look closely, it should be obvious which side the metal pads are on.





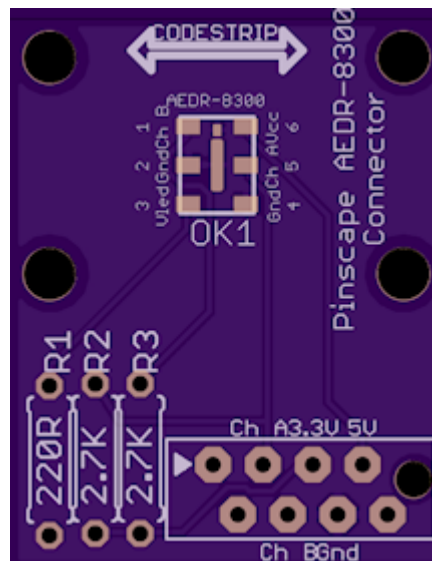
*Front of the AEDR-8300. The little circular areas are the light emitter and detector.
There's a very subtle "chamfer" near pin 6, top right.*



Back of the AEDR-8300. The metal pads for soldering are on this side. The center pad has a little dot, like the dot over an "i", at the "top" end.

Once you find the front and back, all that's left is to find the right rotation to match the circuit board. There are two ways to identify the right rotation. One is to look for the "chamfer" on the front face of the chip. This is a very subtle indentation, not quite a notch. You should be able to see it with a strong enough light. The chamfer is at the "top right" corner, near pin 6. The other way to figure the orientation is to observe the center metal pad, which you can see through the plastic even from the front. When the chip is oriented correctly, the metal pad will look like a lower-case "i", with the little dot at the top. Refer to the diagrams above and look for that lower-case "i" shape.

Orient the circuit board as shown below, and you'll see that the "i" shape on the chip should match the "i" shape in the pad area for the chip on the board (labeled "OK1").



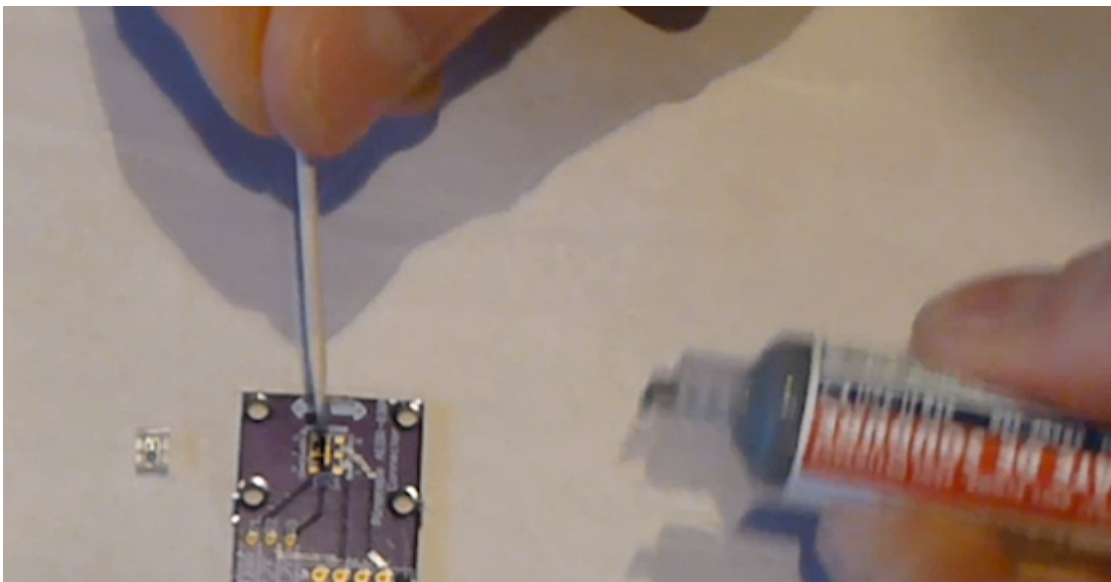
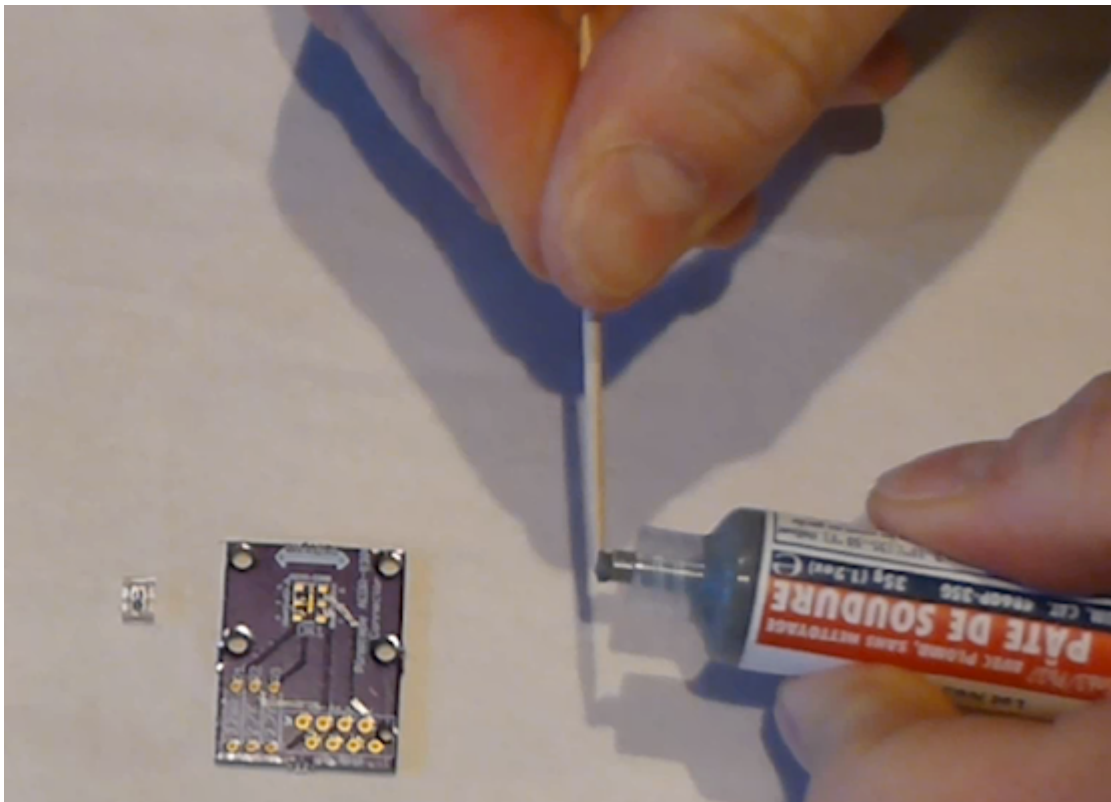
The thing that makes soldering this part tricky is that, as you can see above, all of the solder pads are on the bottom of the chip. That makes it hard to get your soldering iron into contact with the pad to melt the solder.

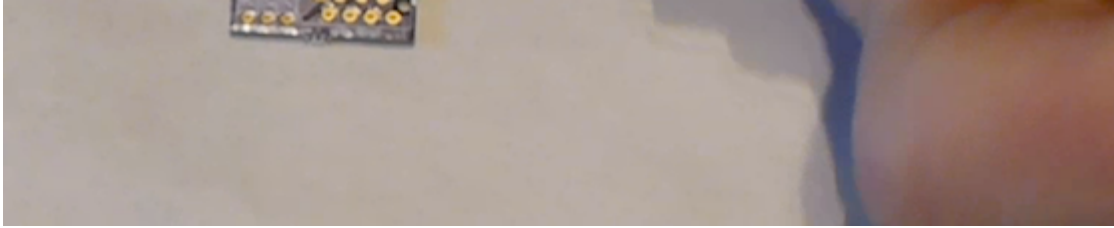
The solution is to use something called "solder paste" rather than ordinary solder, and heat the whole board at once rather than trying to heat the pads individually. This is surprisingly easy even if you've never done it before.

Solder paste is a special mixture of solder and glue that you can spread onto the pads like Cheez Whiz. Regular solder is solid at room temperature, but solder paste is a tacky goo - very much like paste, as the name suggests.

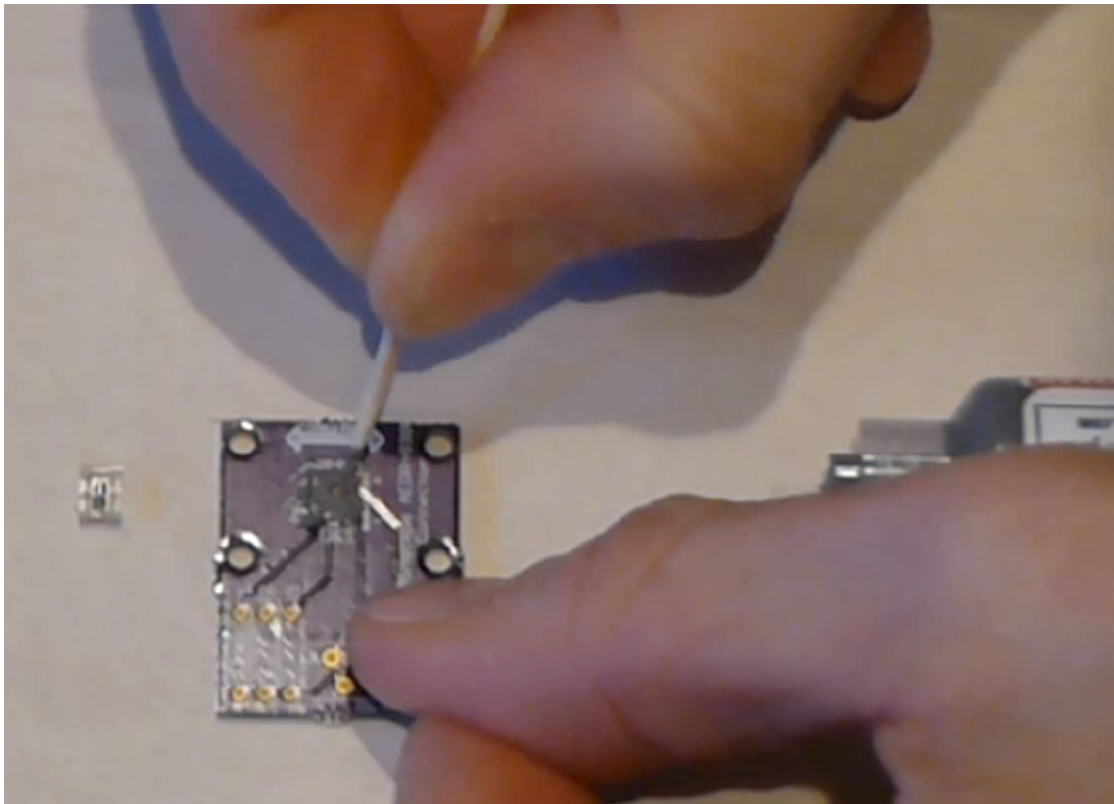
Here's the solder paste I use: MG Chemicals 4860P-35G.

The first step is to smear the paste onto the pads. If you use the MG paste, it comes in a syringe with a needle dispenser. The pads on this chip are so small that even this needle is too big. So I skipped that and used a toothpick to take a tiny bit out of the nozzle (without the needle attached) and smear it onto the circuit board pads.



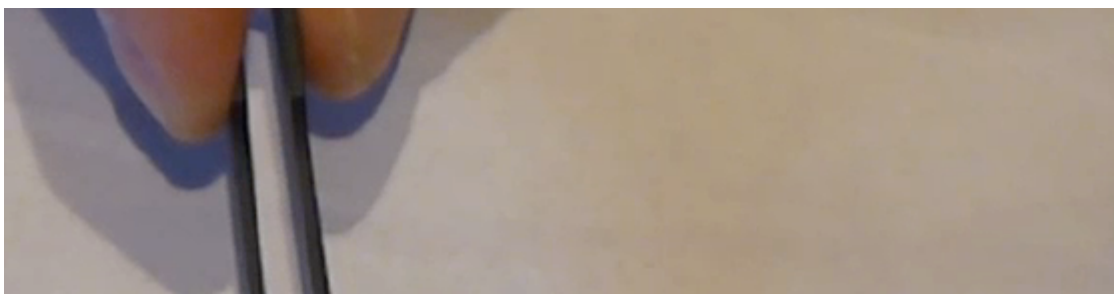


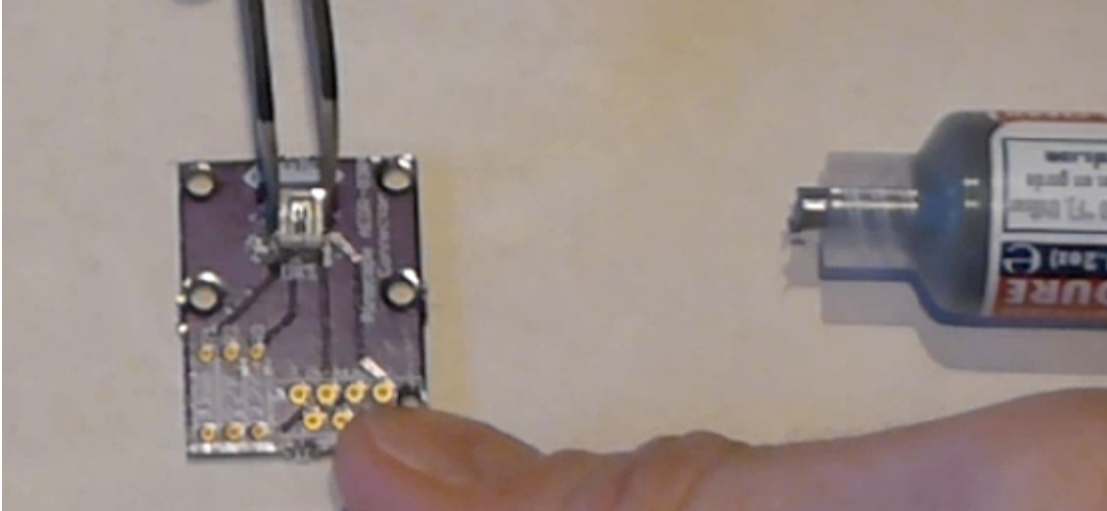
Ideally, you'd put a little bit of paste on each pad. But again, these pads are just too tiny. It's hard to confine each dab of paste to just the pads. I ended up smearing the paste all over the pad area.



It might look like a useless mess at this point, but it's actually okay. The solder paste will come to the rescue when heated. It's chock full of the magical "flux" chemical that makes the solder stick only to the metal parts when melting. The surface tension pulls the excess solder out of the gaps between the pads. The mess cleans itself up. Just make sure the coating of paste is as thin as you can make it. If there's too much solder overall, even the flux won't be able to confine the solder to the pads.

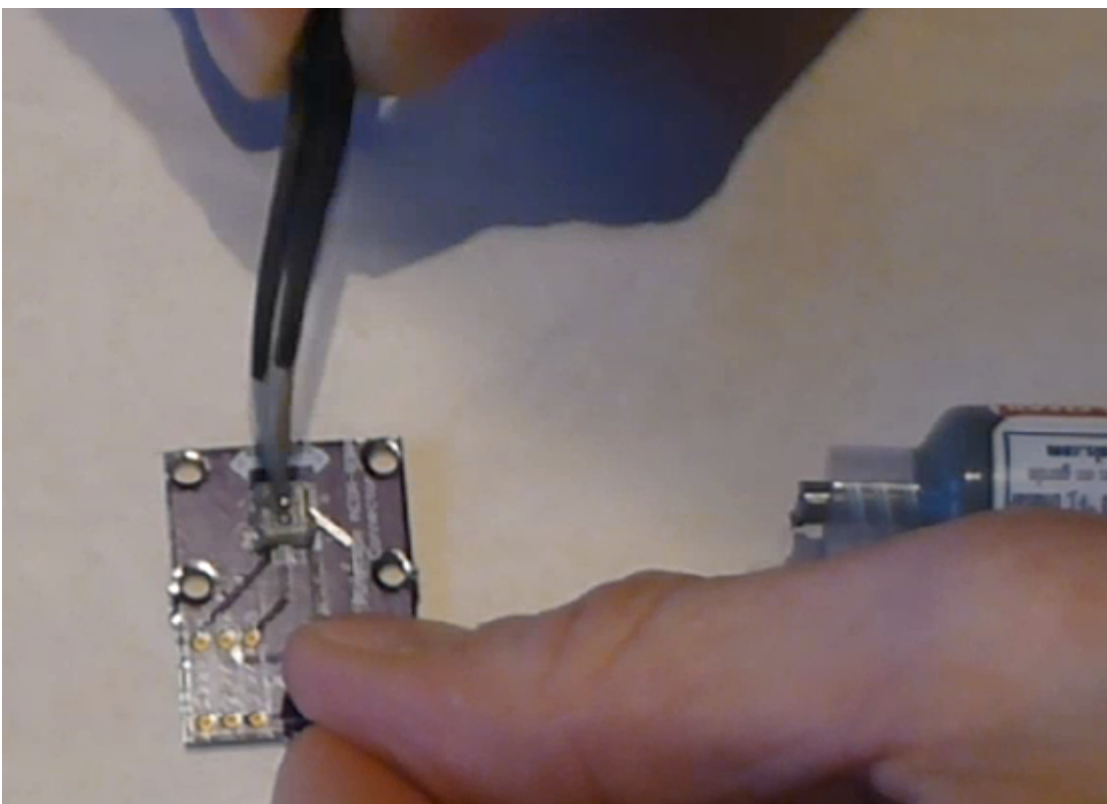
Once the pads are covered in the paste (whether or not you managed to keep it away from the spaces between the pads), it's time to stick the chip onto the paste. Hopefully you already figured out how the chip is supposed to be positioned, and you kept the chip standing by in the proper orientation, so now it's just a matter of popping it into place. Forceps are very helpful at this stage.





You don't have to get the positioning exact. Again, the solder paste will make up for a lot of inexactitude in your prep work, in this case because the surface tension between the solder and the metal pads will actually pull the chip into almost perfect alignment when the solder melts. But you have to be sure that each pad on the chip is at least close to its proper destination pad on the board. The surface tension will pull each pad on the chip to the closest pad on the board. If the chip is badly misaligned, the closest pad might be the wrong one, and the chip might get pulled into the wrong position. So a little care is required here. Just make sure the chip is lined up with the outline printed on the board.

Once it's all lined up, give it a little push with the forceps to make sure it's snugly seated in the paste. Double-check that you didn't dislodge it from proper alignment.



The final step is to heat the board to melt the solder.

The professional tool for this step is an SMD heat gun. ("SMD" stands for Surface Mount Device, which is the kind of chip we're working with here that mounts onto pads on the board rather than via wires that feed through holes.) If you already own

an SMD heat gun, I'm going to assume you do a lot of SMD chip work and know exactly what you're doing, so I'll leave you to it.

If you don't own a heat gun, I can recommend two good DIY alternatives:

- Buy a cheap heat gun at a hardware store. Hardware stores and home centers sell cheap heat guns made for miscellaneous household tasks like stripping paint and heat-shrinking plastic wrap. You can find basic models for as little as \$10-15. For example, Harbor Freight Tools has a \$12 model that works well. These cheap household heat guns don't have precise temperature controls like the ones made for SMD work, but we don't actually need much precision for soldering just one chip.
- Use a toaster oven. You should only do this if you have an old one that you no longer use, because the chemicals in the solder are toxic enough that you shouldn't prepare food in the oven after using it for this. It's also best if your oven uses a quartz or infrared heating element, since these come up to a desired temperature very quickly, which make the process more likely to succeed.

I like the heat gun approach better. It's cheap and it's easier to control.

Whichever route you go, do this in a well-ventilated space. The solder paste makes quite a stink when heated, and releases some volatiles that can irritate your eyes and lungs. You won't want to breathe this in concentrated form.

Using a cheap heat gun

During this step, you'll want to use something other than your hands to hold the board in place while you work, since it will get quite hot. You can tape it down to a piece of plywood, or use tongs, for example.

Throughout the heating process, I recommend keeping the heat gun pointed at the chip, but move it around slowly in small circles to even out the heating.

The first step is to warm up the board for about 2 minutes at low heat, to about 250°F. We want to get it warm, but not hot enough to melt the solder. The goal is to warm everything up gradually, so nothing jumps around when we turn the heat up to soldering temperatures.

If your heat gun has multiple temperatures, use the low setting for this first phase. Cheap heat guns don't usually have exact temperature settings, but you might at least have high/low settings. Many of the cheap guns have a nominal low setting of 600°F or so, which is higher than we're after at this stage. If yours is like this, just hold it back about six inches from the board so that the board doesn't get the full heat initially. Monitor the solder paste visually during this phase and make sure it doesn't start melting; if it looks like it's liquifying, back off further with the heat gun. If the board isn't even getting warm, move the gun in closer.

Once the 2 minutes is up, the next phase is to increase the heat enough to melt the solder. We want to heat the board to about 500°F at this point. Again, many cheap heat guns don't have a setting this low, so you might not even need to switch settings at this point, but simply move the gun closer to the board.

This step should be fairly quick. Again, visually monitor the solder paste. If your heat gun temperature is high enough, the paste should begin to liquify within 15 seconds or so. You should see it start to run and bubble. Shortly after that starts, the paste will transform from the dull gray you've seen so far to shiny metal. That's the solder; the flux that was mixed in is separating from the solder and evaporating, leaving behind the shiny solder. If the paste doesn't melt and turn shiny within 30 seconds,

turn up the heat or move the gun closer.

Maintain this heat level for about 10-15 seconds after the paste has all transformed into solder, then turn off the heat. You want to give it long enough for the solder to melt evenly and adhere to the pads on both the chip and the board. You should see the chip settle in closer to the board as the surface tension of the melting solder spreads the solder out across the pads.

Allow the board to cool for a few minutes.

Using a toaster oven

As we mentioned above, only use a toaster oven that you don't use for cooking food. The solder paste contains toxic chemicals. You shouldn't use it to prepare food after this since the chemicals could leave some residue in the oven.

It's best if your oven heats up very rapidly, because the timing of the temperature phases is fairly important. Quartz or infrared heating elements are great for this because they heat up almost instantly. If your oven takes a while to pre-heat, one suggestion I've seen is to use *two* ovens, one for the low-temperature phase and the other for the high-temperature phase. That way you can pre-heat both ovens to the correct temperatures, and move the board from one to the next at the proper time.

Some people also recommend a skillet on the stove top, but I haven't tried that.

The basic plan is to heat the board in three steps. The first step is a pre-conditioning phase at medium temperature. This is called the "soak" phase in manufacturing lingo. The goal is to get everything thermally stable near but below the solder melting point, so that nothing jumps around due to thermal shock during the melting phase. The second step is at high temperature, where we actually melt the solder. The last step is to turn the heat off and let the board cool off gradually.

Step 1: 250°F for 2 minutes.

Step 2: heat to 500°F (or as hot as your oven gets; 450°F works for the MG paste). Watch the board carefully at this stage: after about 10 seconds, you should see the solder paste start to change from gray to shiny silver as the solder melts. After about another 10-20 seconds, you should see the IC chip move slightly - it should look like it's getting sucked into position. It should straighten up and get visibly closer to the board as the surface tension draws the solder to the pads and pulls the chip pads close to the board pads. Once this happens, give it a few more seconds to make all of the solder is melted, then move on to the next step. Don't stay at full heat for more than about 60 seconds, as you don't want to overheat the chip.

Step 3: turn off the heat and let the board cool in place for about 30 seconds. Then open the door to let it cool faster. You can take the board out after a couple of minutes. Use tongs or gloves, as it could still be hot enough to burn you.

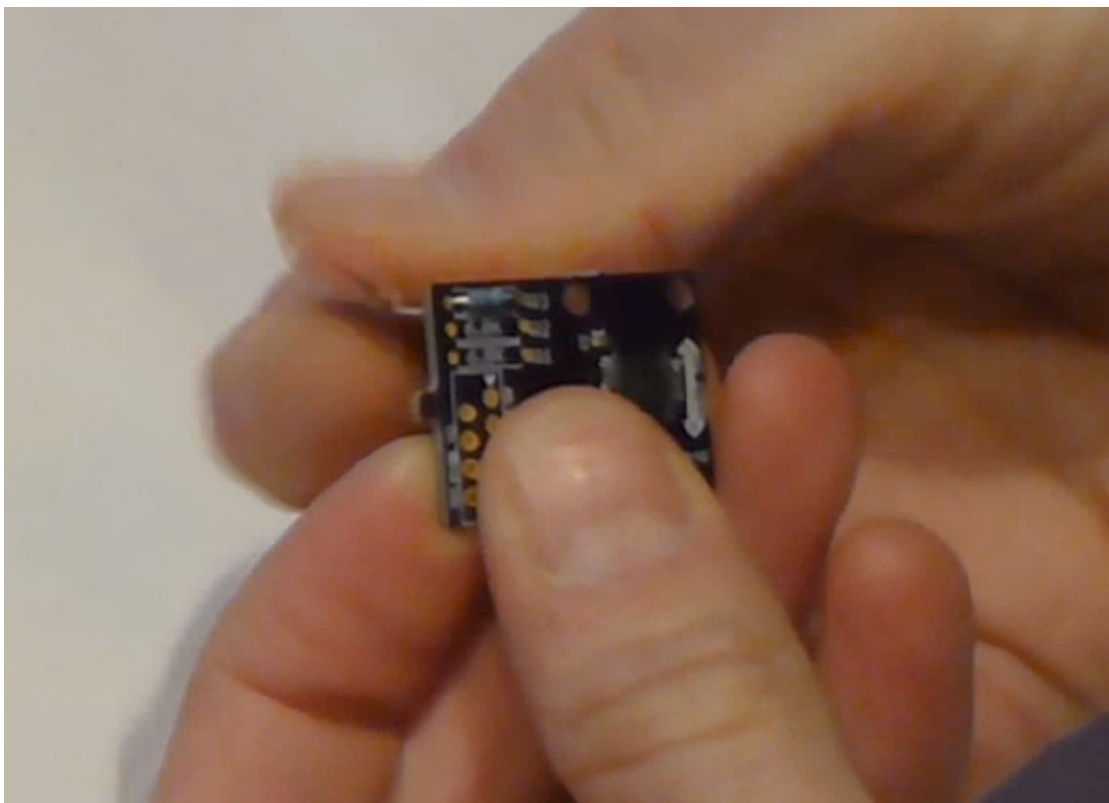
Check the result

Once the board, cools, visually inspect the result. Make sure that the chip ended up in the right place. The solder should have pulled it into the right position rather than away from it. The clear package on this particular chip makes it possible to see through to the pads, which is a big help. You should be able to see that the solder has migrated out of the spaces between the pads.

Install the resistors

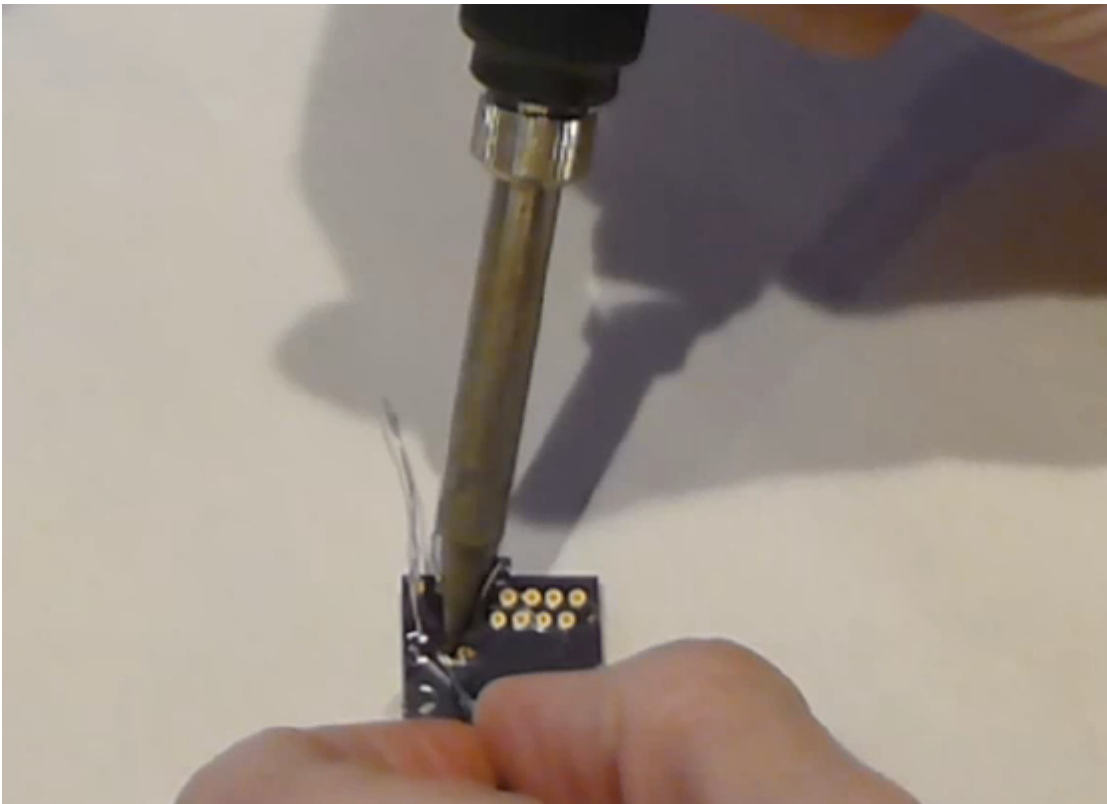
Solder the three resistors using conventional soldering techniques.

For each resistor, bend the leads at right angles and insert them through the marked holes, with the resistor body on the top of the board (the side with text printed). Make sure you put the right resistor in each slot; the resistance value for each one is printed on the board right where it goes. Resistors aren't polarized, so it doesn't matter which direction it goes. Push the resistor body or pull the leads until the body is flat against the board.



Turn on your soldering iron. Turn the board over. Hold the tip of the soldering iron against the point where the resistor wire and board pad meet. The goal is to heat up both pieces of metal, hot enough to melt the solder. Give it a few moments to heat

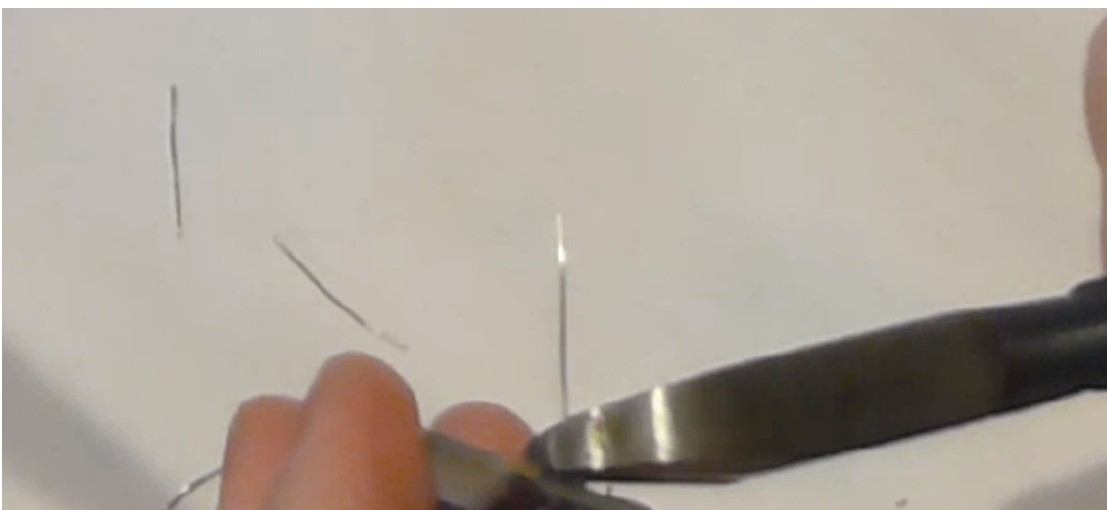
up, then touch the solder to the lead/pad junction point you're heating. Let the solder melt and flow over the joint, then remove the iron. Keep everything perfectly still for about 5-10 seconds until the solder fully hardens.



The big rookie mistake in soldering is to focus on the soldering iron tip when applying solder. What you really want to do is apply the solder to the metal parts you're trying to join - the resistor wire and the board pad. Those parts need to be hot enough to melt the solder on their own. That makes the solder flow onto the parts and stick to them as it cools. If you apply the solder to the soldering iron tip, it won't flow properly onto the metal parts you're joining and won't form a good joint.

Visually inspect the solder joint after you're done and make sure the solder evenly covers the pad and resistor wire without any gaps. If you see any gaps, the solder might not have flowed properly onto all the metal, so re-heat it and apply more solder if necessary. You can also wiggle the resistor and make sure the wire doesn't move at all in the joint.

Now you can trim the resistor wire with wire cutters. Cut off the excess lead past the solder bubble.

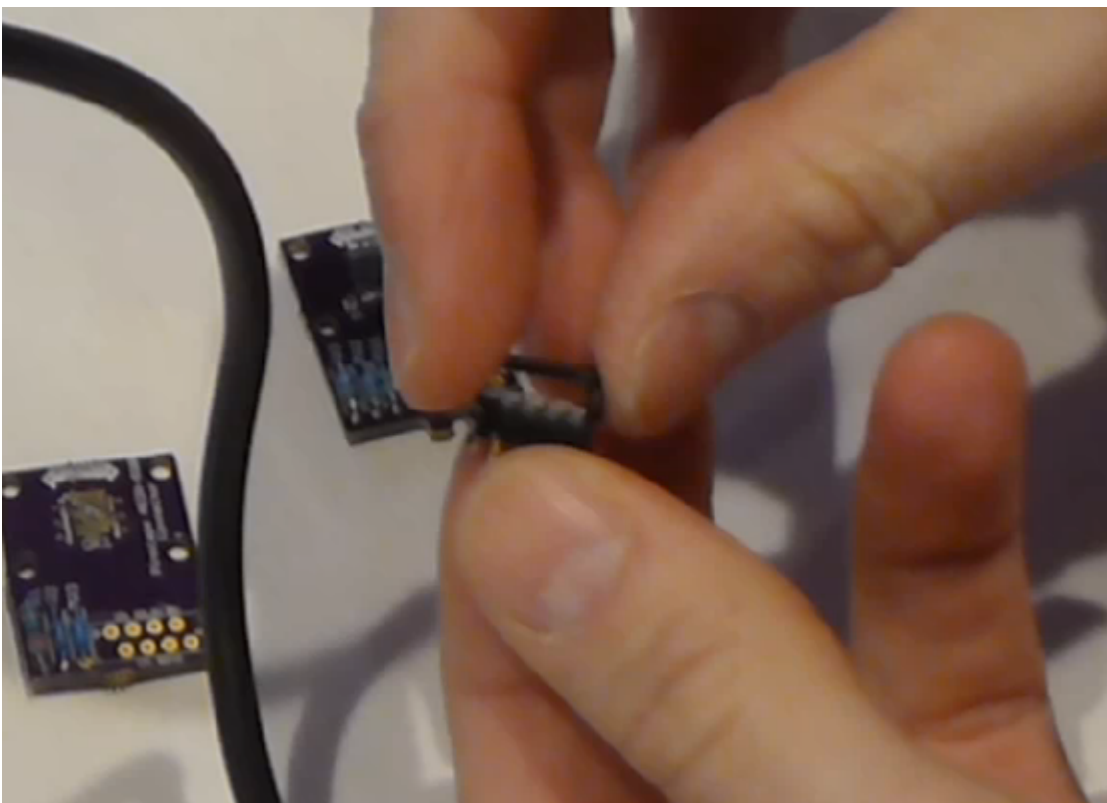




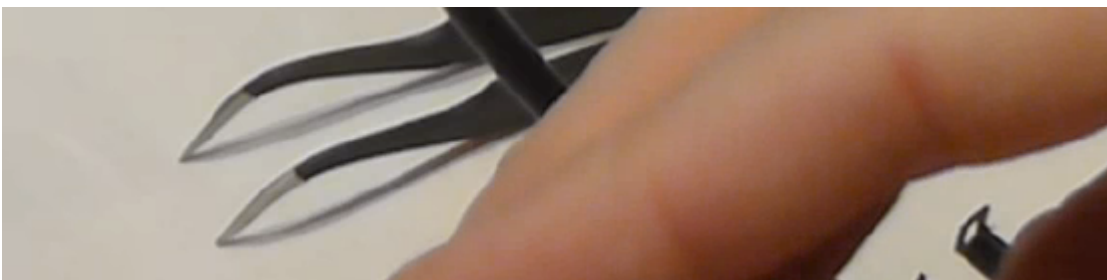
Ribbon cable wiring

For the expansion boards, you should install the ribbon cable connector from the parts list. The connector matches the pin layout of the plunger header on the main expansion board, so you can just plug it in directly once you assemble the cable.

The connector comes in two pieces: a base with the pins sticking out, and a clip that fits over the top. Separate these for now and set the top clip piece aside. Be careful handling the bottom piece: the pointy spikes sticking out of the top are sharp. They're basically little wire cutters designed to cut through the cable insulation on their own, which we'll come to in the final step.



Pop the connector onto the board through the marked holes. Note that it only fits in one direction: there's a little plastic peg on one side that has to fit into the corresponding hole on the board. If you can't get the connector to fit properly, try rotating it 180° in case you have the peg on the wrong side.

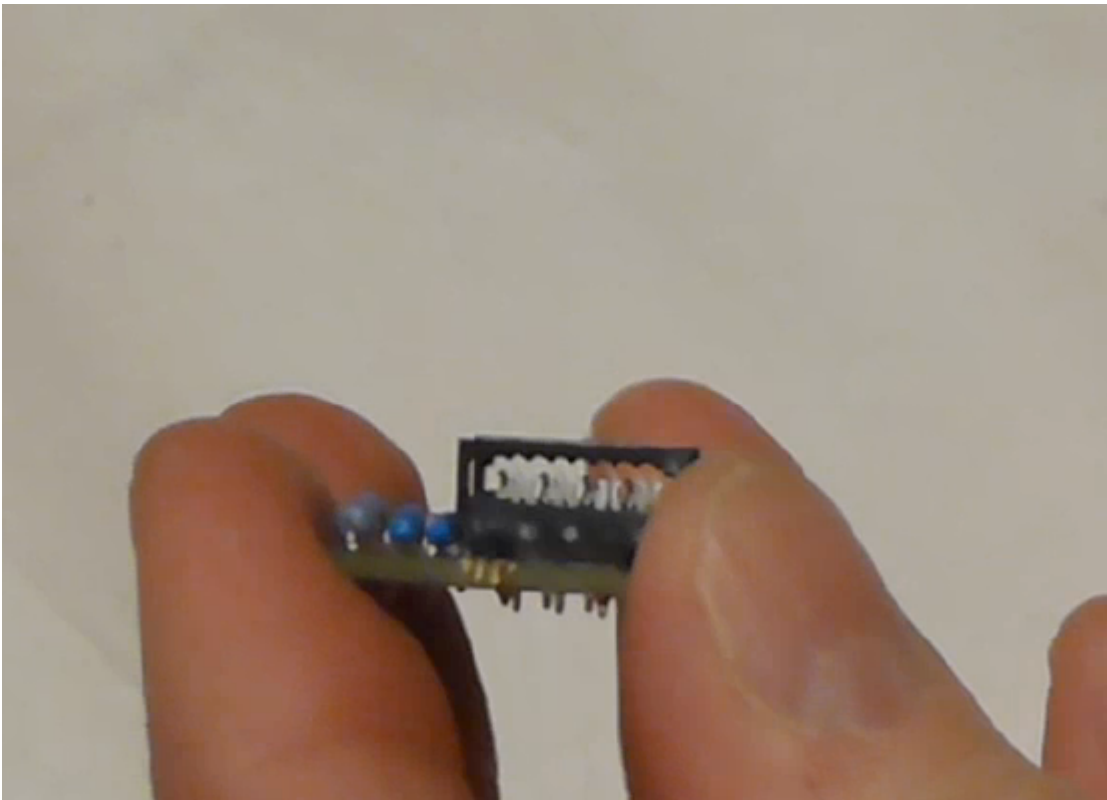




Flip the board over and solder the pins from the back side of the board. This is just like soldering the resistor leads. (Except that there's no excess wire to clip in this case.)

Now it's time to attach the cable. This connector is of the "IDC" type, which stands for Insulation Displacement Connector, which means that it's designed to pierce the cable insulation when you press the cable into it. These connectors are designed to do most of the work for you, so don't worry if you haven't done this before.

Grab the top "clip" piece for the connector that you set aside earlier. Position it **loosely** on top of the base. You can see that it slides into latches on either side. Don't push it down all the way yet; leave a gap big enough for the cable.



Now slip the cable into the gap. If your cable has a stripe (usually red) down one edge, put that on the side with the triangle/arrow printed on the circuit boards - that's pin #1. The red stripe will make it easier to identify the corresponding pin #1 at the other end. If your cable doesn't have a stripe, I'd strongly recommend adding a stripe with a red marker. Use an **oil-based ink** marker - water-based inks won't

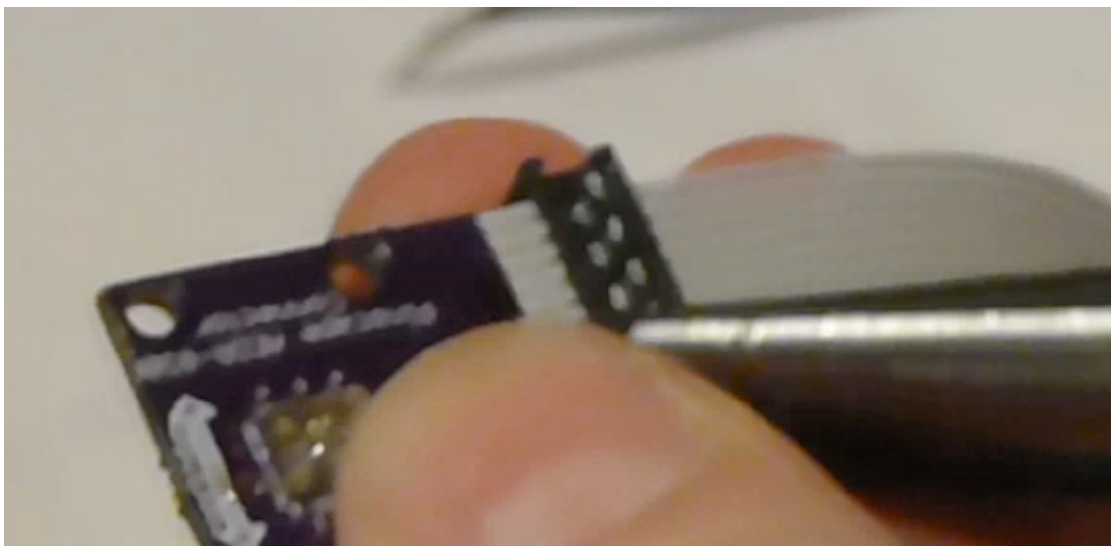
stick to the plastic insulation. Draw the stripe down the whole length of the cable along one side (it doesn't matter which one; you just pick one as the "pin 1" side).

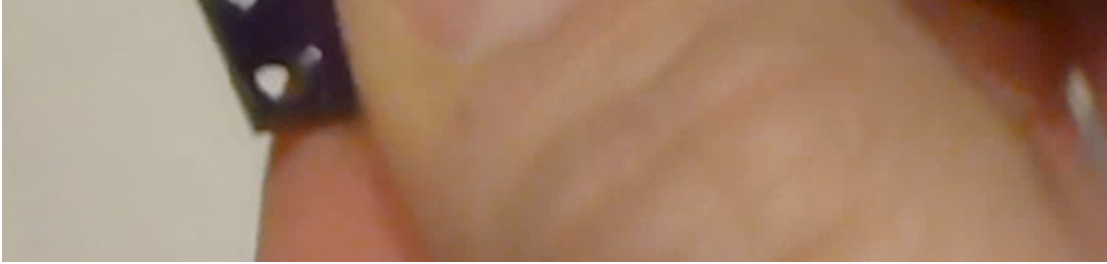
The cable should just barely fit into the gap. This is part of the design, to ensure that the cable is positioned properly. Let the end of the cable extend about 1/4" past the clip.

Once you have it positioned properly, get out some pliers. (There's a specialized IDC crimper tool for this job, but ordinary pliers will work if you're careful.) Carefully apply pressure to the top of the clip. Start at one end, push it down just a little bit there, then gradually move to the other end. Work your way back and forth a few times until the clip is all the way down and snaps into the locks. You have to be careful not to do this all at once, since the locks aren't strong enough if the pressure is too lopsided.



When you're done, the clip should be fully flush with the sides locked into the latches. You'll be able to see the insulation poking into the holes on the top of the clip.





Finally, install the IDC connector at the other end. This is the plug that connects to the expansion board header. This is almost exactly the same as assembling the first connector; the only difference is that there's nothing to solder this time.

The one thing to be careful about is to line up pin #1 on the plug with the pin #1 wire in the cable. On the plug, you should find a small triangle or arrow at one corner. That's the pin #1 side. If you already identified pin #1 with a red stripe on the cable, make sure the stripe is on the pin #1 side of the plug. Pin #1 on the sensor board corresponds to pin #1 on the expansion board plunger header, which is marked on the expansion board with an arrow. Just line up the pin #1 markings down the whole chain and everything will communicate properly.

Standalone KL25Z wiring

If you're using a **standalone** KL25Z (without the Pinscape expansion boards), wiring is a little tricky, because the pins you have to connect it to on the KL25Z are scattered around different pin headers.

I recommend using a ribbon cable and the Chapter 100, plunger sensor breakout board to connect to the standalone KL25Z. It'll make things much easier in the long run by giving you a pluggable connector between the plunger and KL25Z.

- Build the ribbon cable connector exactly as described above, as though you were using the expansion boards
- Follow the instructions in Chapter 100, Plunger Sensor Breakout Board to build the breakout board
- Connect the following wires between the breakout board and the KL25Z:
 - Breakout board **5V** to KL25Z 5V (pin 10 on J9)
 - Breakout board **3.3V** to KL25Z P3V3 (pin 8 on J9)
 - Breakout board **GND** to KL25Z GND (pin 12 or 14 on J9)
 - Breakout board **D0** to KL25Z PTD0 (pin 6 on J2)
 - Breakout board **D5** to KL25Z PTD5 (pin 4 on J2)

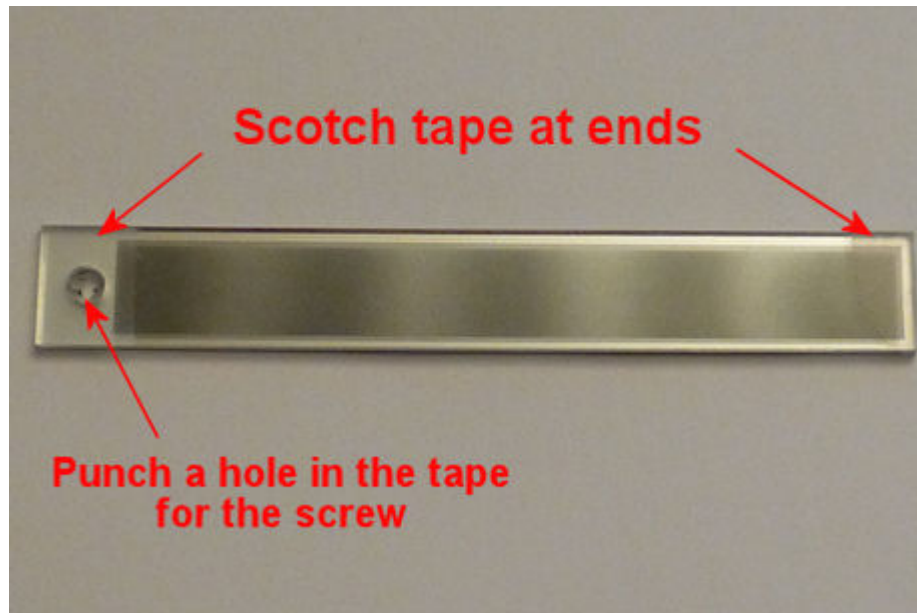
If you want to use ad hoc wiring instead (which I don't recommend), see "Plug it in" below for wiring instructions.

Final assembly

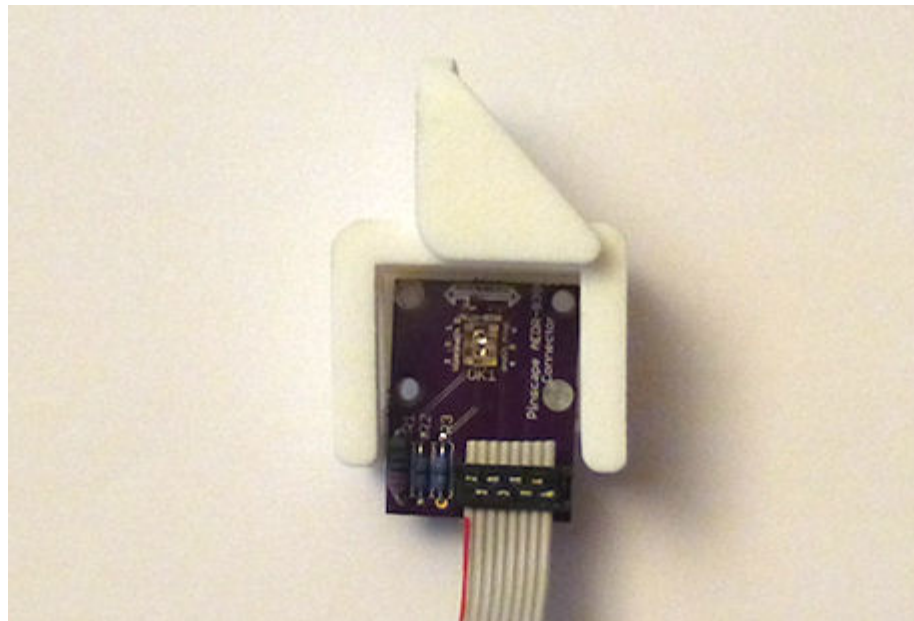
Good news! The circuit board was the hard part. The rest is almost easy.

Laser-print the scale graphic (see the parts list at the start of this chapter) on transparency film. Cut out the bar area to just slightly smaller than the acrylic guide rail (so that the edges don't overhang). Attach it to the acrylic. Attach it on the shiny mirrored side, with the printed side facing the acrylic (this will help protect the printing from wear as the sensor slides back and forth). You can just use a little piece of Scotch tape at each end to attach it, making sure it's pulled tight so that it stays

flat against the acrylic.



Attach the circuit board to the 3D-printed sensor bracket. Place it with the sensor facing outward and the ribbon cable or wires at the bottom. Secure it with four small machine screws and nuts. I recommend M2x8mm or M2x10mm, or #2x $\frac{3}{8}$ ". Nylon parts are ideal here. Insert the screws from the component side of the board so that the nuts are on the back of the bracket.



If you already fully installed your plunger, I'm afraid you're going to have to take it back apart at this point. Remove the e-clip that's holding the main spring in place (a pair of pliers is helpful: first hold the spring back so that it's not pressing against the clip, then grab the back edge of the clip with the pliers and pull it off). Slip the spring off. Now remove the **top two screws** from the housing. Leave the bottom screw in place.

If you haven't already installed your plunger, it's time to do that. Insert the housing through the opening in the front of the cabinet. Slip the mounting plate over it, aligning the screw holes. Screw the **bottom screw only** into the housing and tighten. Leave the top two screws out for now. Slip the barrel spring onto the shooter rod, then add a washer. Make sure the nylon sleeve is installed in the housing, then

slide the rod into the housing. Add the second washer on the inside.



Slip the 3D-printed scale bracket over the shooter rod. It fits over the shooter rod holder in the housing, and the screw holes align with the screw holes in the housing.



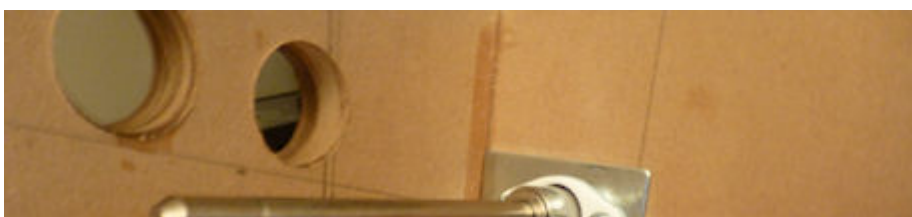
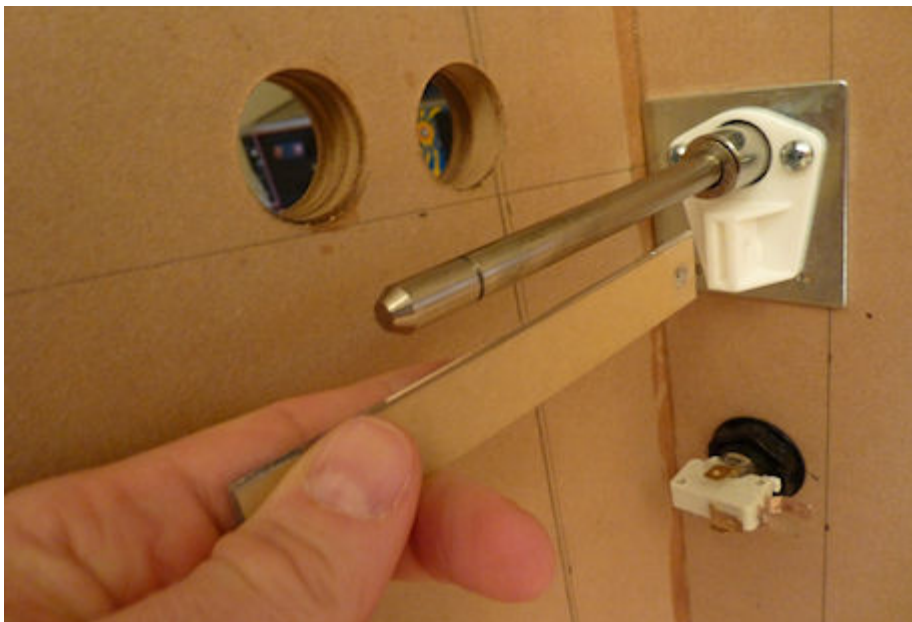


Install the two screws. Don't overtighten, to avoid stressing the plastic.



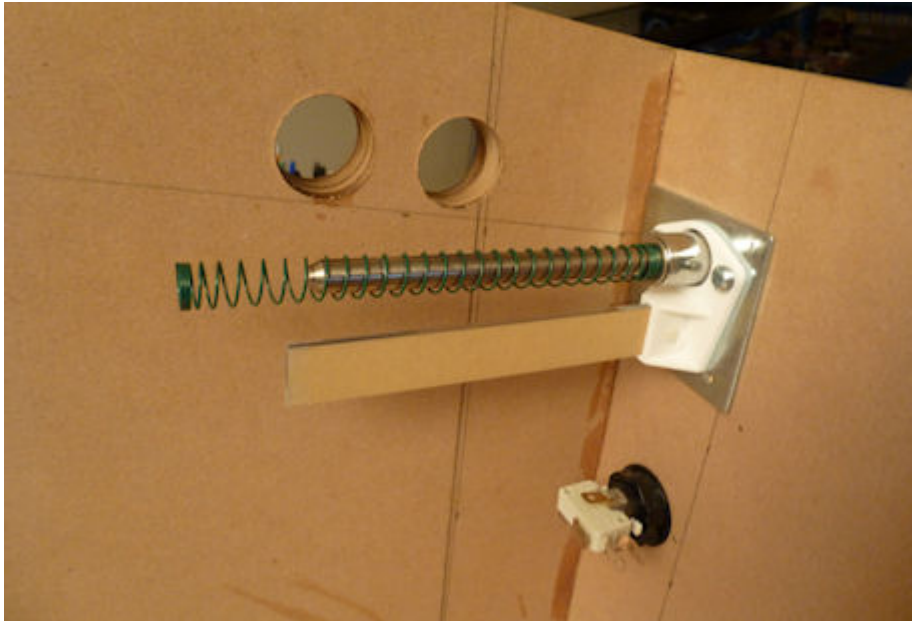
Pop the scale/guide into the slot in the bracket, with the printed side facing the cabinet wall.

Fasten it with a small bolt and nut through the provided hole in the bracket. Any nut/bolt that fits will work; an M3x12mm or #4x½" should work well. I'd recommend a nylon bolt and nut if you have them handy. In any case, don't overtighten; this one doesn't have to handle much force, so just make it tight enough that it won't work itself loose.

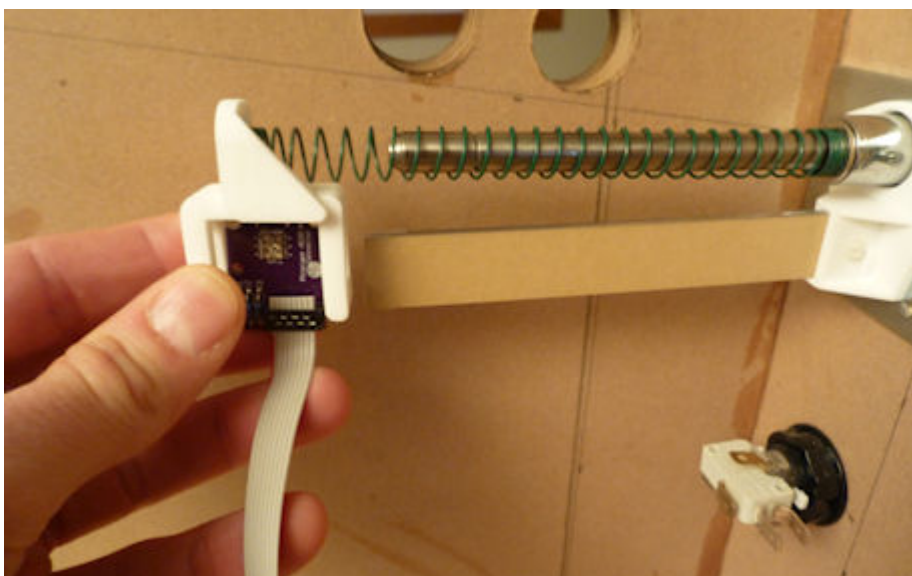


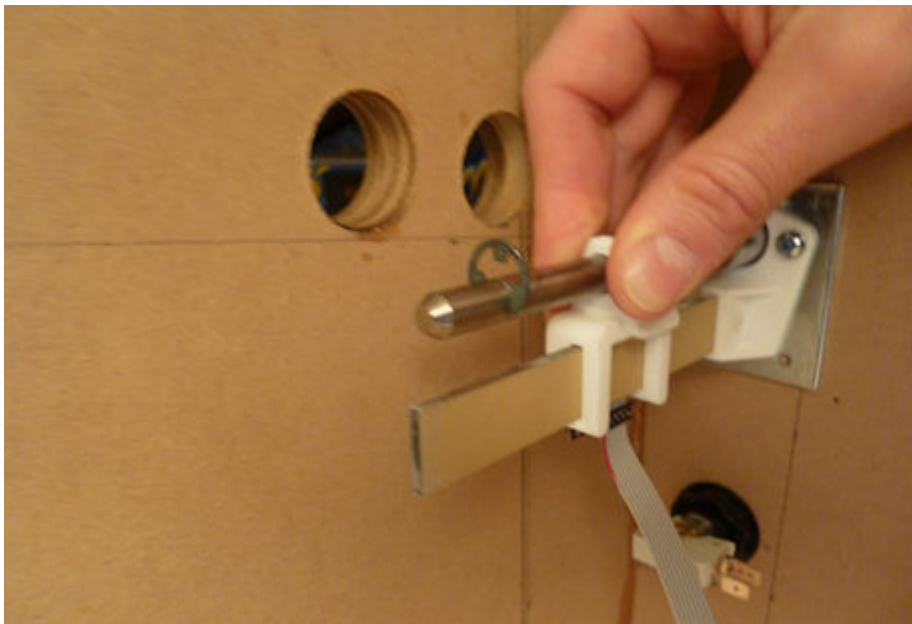
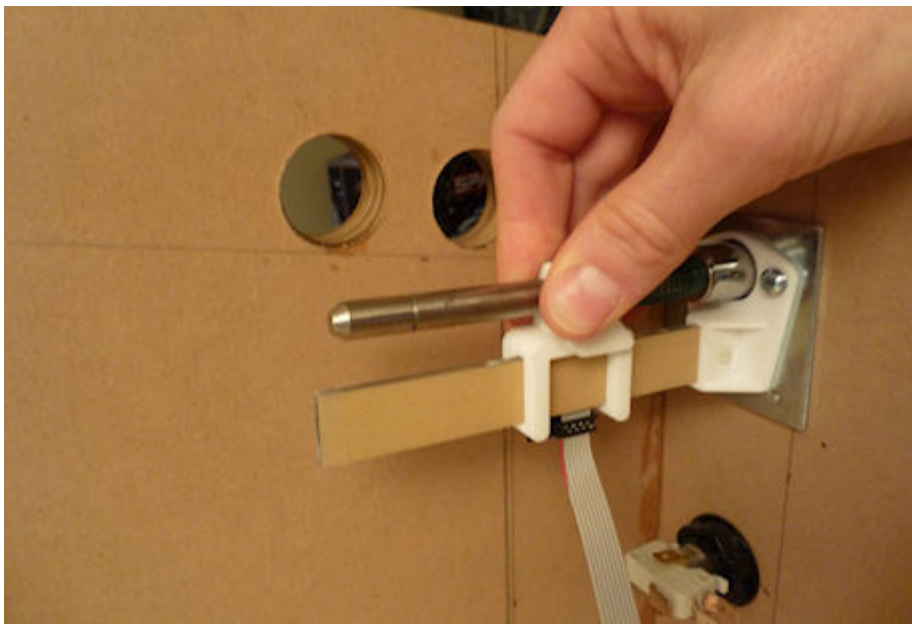
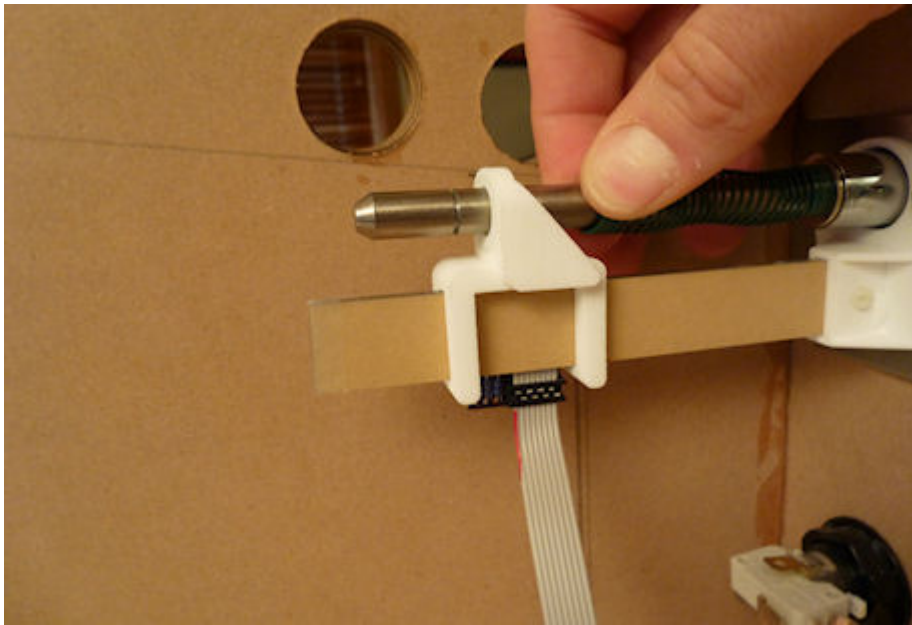


Slip the spring onto the shooter rod.



Slide the assembled sensor bracket onto both the shooter rod and the guide rail. The round hole in the top fits over the shooter rod and the slits fit over the guide rail. You'll probably want to compress the spring with one hand while sliding the bracket onto the rod. Once it's in place, keep holding the bracket back (compressing the spring) and slip the e-ring into its slot on the shooter rod. Pop it into place with a pair of pliers. You can now gently release the spring tension so that the spring pushes the sensor bracket against the e-clip. This is the final working configuration.

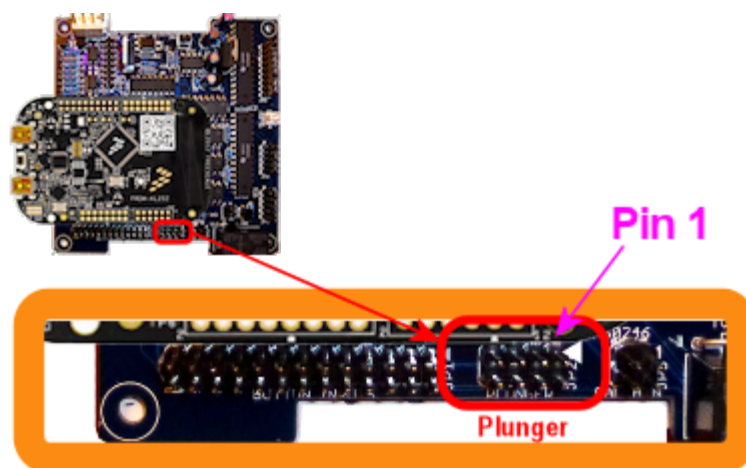




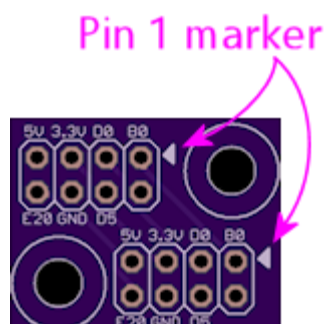


Plug it in

Expansion board: Plugging the sensor into the expansion board is easy if you used the ribbon cable connectors. Just plug the 8-pin connector into the PLUNGER header on the main board. Make sure pin 1 on the plug corresponds to pin 1 on the board, which is marked with a little white triangle printed next to the header.



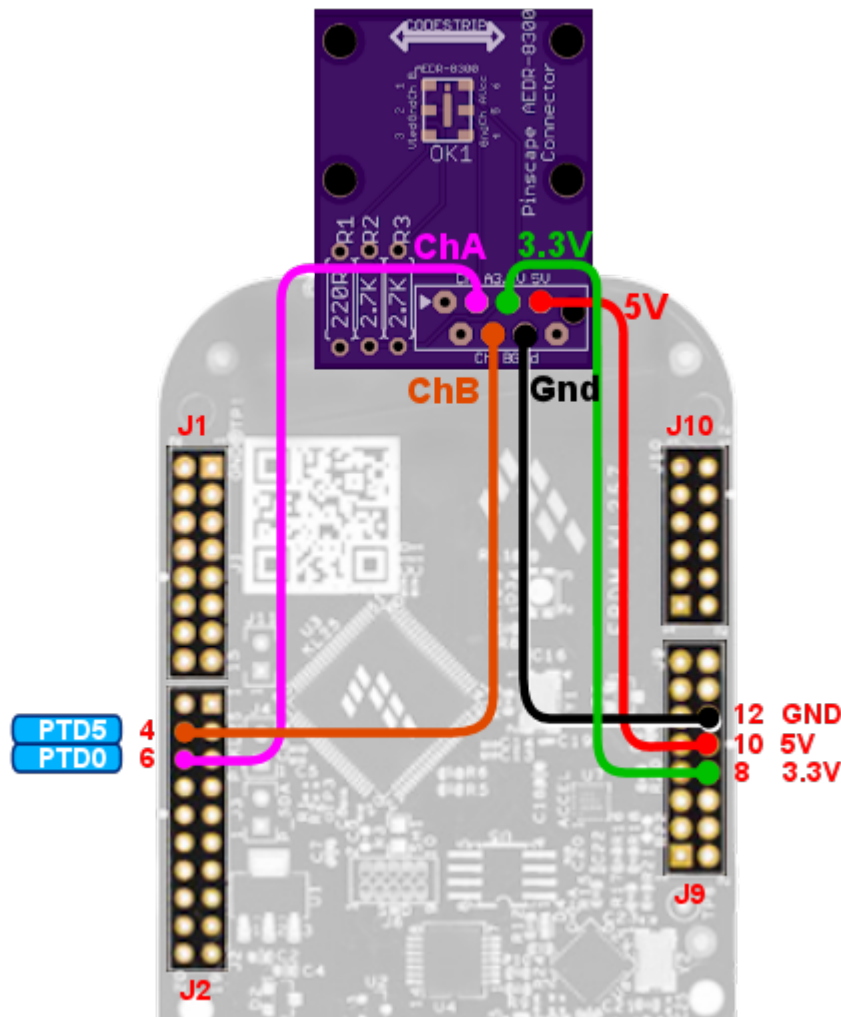
Standalone KL25Z: If you're using the plunger sensor breakout board as recommended, just plug the ribbon cable connector into the header on the breakout board. Be sure pin 1 on the plug corresponds to pin 1 on the board, which is marked with a little white triangle printed next to the header.



If you prefer to use ad hoc wiring - which I don't recommend - you can just run

some hookup wires between the sensor board and the KL25Z. Be sure to make the wires long enough to reach comfortably between the sensor and KL25Z, and remember to account for how the sensor moves with the plunger. Follow the wiring plan below.

Sensor Board Pin	KL25Z Pin
3.3V	P3V3 (J9-8)
5V	P5V (J9-10)
Gnd	GND (J9-12)
Ch A	PTD0 (J2-6)
Ch B	PTD5 (J2-4)

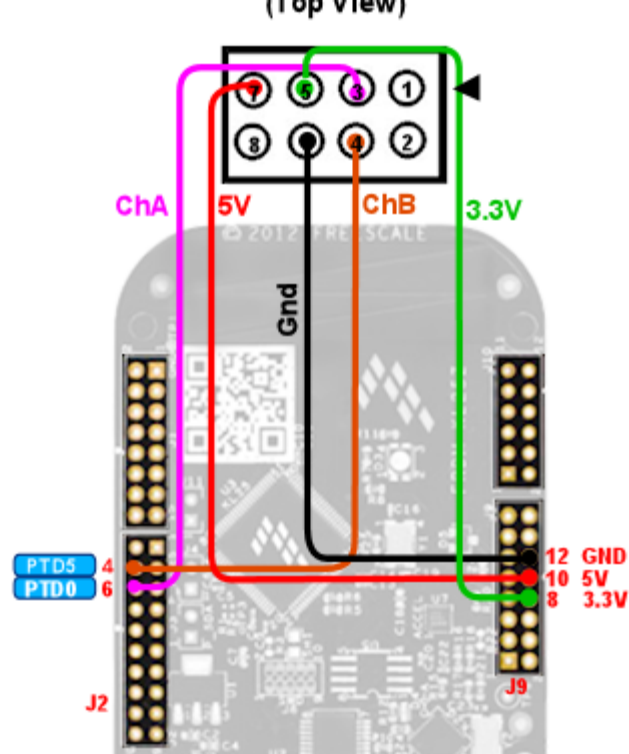


Note that the two GPIO ports, PTD0 and PTD5, are only suggestions. If you're already using these ports for some other function, you can assign the sensor inputs to other ports using the Config Tool. However, note that only ports with "PTA" or "PTD" prefixes can be used for these. (The inputs have to be PTA or PTD ports because only those ports are capable of generating interrupts on the KL25Z. The Pinscape firmware needs interrupt capability on the inputs to process the signals from this sensor.)

(The power and ground wires aren't configurable. Connect those as shown.)

In case you want to set up your own custom connector for the ribbon cable, here's how the pins on the expansion board end of the ribbon cable connector are arranged:

Expansion Board Connector
(Top View)



Software setup

If you haven't already set up your KL25Z with the Pinscape firmware, you'll need to do that first. See Chapter 89, KL25Z Software Setup.

Start the Pinscape Config Tool. Click the Settings button for your device. Scroll down to the Plunger Sensor section. In the Sensor Type drop list, select AEDR-8300.

(If the AEDR-8300 option isn't available in the plunger sensor list, you probably have an older version of the Config Tool. Updating to the latest version should add the option.)

If you're using the expansion boards, the pins should be configured automatically. If you're using the standalone KL25Z, select the pins you wired to the sensor's data channels ("Ch A" and "Ch B" on the sensor board).

I recommend enabling the "auto-zeroing" feature, and setting a fairly long delay time, perhaps 60 seconds. If this feature is enabled, the Pinscape firmware will "zero" the plunger when it hasn't moved at all in the amount of time you specify. Zeroing means that the firmware assumes the plunger is exactly at the normal rest position. Why do this? Because the AEDR-8300 is a purely "relative" position sensor. That means it doesn't ever know the plunger's true position; it only knows how far it's moved since the system was turned on. If the sensor ever misses a tiny bit of physical motion, the sensor's notion of the relative position will get a little out of sync with the true position. Auto-zeroing corrects for this by forcing the internal position counter back to the starting position whenever the plunger is motionless for a long time. It's usually a safe bet that a perfectly motionless plunger really is sitting at the normal rest position, since the spring always takes it back there when you're not intentionally moving it. Just be sure to pick a long enough time that you won't ever hold it still that long during normal play, such as when lining up a skill shot. 60 seconds seems like a good choice, but use your discretion if you think that might not be long enough. You can also disable this feature entirely if you ever find it troublesome. In my own testing, the AEDR-8300 is remarkably close to perfect at picking up every bit of movement, so in practice you might never find that the plunger gets out of sync with reality in the first place.

Save the new settings by clicking "Program KL25Z" at the bottom of the window.

You should now test and calibrate the plunger. Return to the home screen in the Config Tool and click the Plunger icon for the unit with the sensor attached. This will let you look at the raw sensor input. Move the plunger and make sure it seems to be tracking properly.

If the sensor is working properly, click the Calibrate button in the plunger viewer window to begin the calibration process, and follow the on-screen instructions.

If the sensor doesn't seem to be working, go back to the Settings screen and double-check the sensor pin assignments. Make sure that none of the pins are marked with warning icons (⚠️). If you see any warnings, click on the icon for details. In most cases, the problem will be that you've assigned the same pin to multiple functions. If so, go to the other place the pin is assigned, and clear that entry by setting it to "Not Connected".

If the software setup looks okay, check the physical wiring. Inspect each wire and make sure that it goes to the proper pin on each end (KL25Z and sensor board). Check that each GPIO port assignment on the settings page matches up with the physical pin on the KL25Z and connects to the corresponding terminal on the sensor board.

Backwards operation

If the on-screen plunger appears to move backwards from the physical plunger, you can fix it in the software without reinstalling the sensor. Open the Pinscape Config Tool. In the row for the controller, click the Plunger icon. Check the box for "Reverse orientation". (Or, if it's already checked, un-check it.) This tells the software to reverse the readings from the sensor, so that it acts like it was installed in the opposite orientation.

Note: Versions of the firmware released before January 2020 had a bug that made the "Reverse orientation" option not quite work right with this sensor. If you need to use this option, you should update the firmware to a 2020 (or later) version. If for some reason you can't or don't wish to update, then instead of using the "Reverse orientation" checkbox, you can achieve the same effect by going to the Settings page, finding the GPIO pin mappings for the sensor, and swapping the "Channel A" and "Channel B" pin assignments. That will reverse the way the software interprets the directional signals from the sensor, achieving the reversed motion you're after.

105. Plunger Setup (TCD1103)

This is one of the more experimental of the Pinscape sensors, in that very few people have actually built one (as far as I've heard, anyway). It's a relatively high-difficulty project that requires building a custom circuit board with surface-mount parts, with the sensor itself being particularly challenging to hand-solder, since it's one of those fiendish modern devices with the solder pads completely hidden on the bottom of the package. The project also requires assembling a 3D-printed housing and finding the right type of optical lens to focus the image, so when you put all of these pieces together, it can be quite daunting. I've successfully built a couple of these myself, so I can at least assure you that it's doable, but it's definitely a challenging project - or, really, several projects in one, when you consider the electronic, mechanical, and optical elements. The payoff if you can get it working is that this is a really nice sensor that produces excellent, high-resolution, high-speed plunger position readings.

The original Pinscape plunger system, first published back in 2015, was based on a special type of optical sensor called a "linear image sensor", meaning all of its pixels are arranged into a single row. That's in contrast to a regular camera's sensor, which has a rectangular array of pixels. The particular linear image chip used in the original project, known as the TSL1410R, was an improbably perfect fit for the plunger application - and I mean "fit" literally, in that the sensor was almost exactly the same size as a standard plunger mechanism. The sensor consisted of a row of sensor pixels about 3" long, exactly the length of travel of a standard pinball plunger. By placing the 3" sensor window adjacent to the plunger, and placing a light source on the other side of the plunger, the sensor was able to determine the plunger position by scanning the pixels for the shadow cast along the 3" window by the end of the plunger. As you move the plunger back and forth, the shadow moves along that 3" array of pixels, so the software can easily detect the motion by taking snapshots at regular intervals and comparing the location of the shadow from one frame to the next.

Sadly, the manufacturer stopped making that 3" sensor a long time ago, and there's nothing else like it on the market, so this hasn't been an option lately for new pin cab builds. There are some excellent alternatives based on completely different technologies, particularly the Chapter 104, AEDR-8300 quadrature sensor and the Chapter 102, sliding potentiometer design. But none of the other available designs has the special charm of an imaging sensor, which is that it's completely "non-contact", meaning that there's no mechanical connection between the sensor and the plunger, and thus no moving parts to wear out over time. Operating a standard ball shooter involves a lot of force and high speeds, which inevitably causes some wear on the sensor parts if they have to be mechanically connected and move at the same high speeds and jerky accelerations. (There is another decent non-contact option: the Chapter 103, VCNL4010 proximity sensor. But that sensor isn't nearly as precise as the TSL1410R was.)

That's why I'm pleased to present a new option that brings back the non-contact, all-optical approach, using a different sensor that you actually can buy (as of 2024). The design for the new sensor isn't quite the same as the old one, because the new

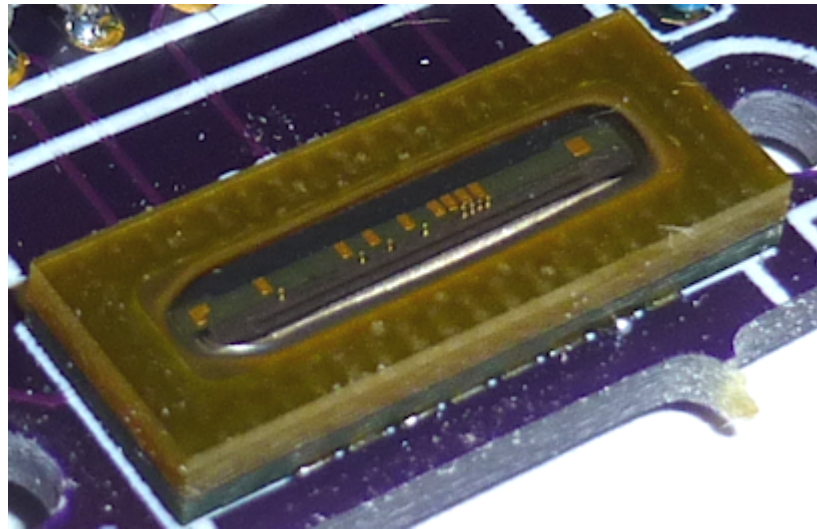


sensor is rather different. Whereas the old sensor had that gigantic 3" pixel array, the new sensor described in this section has a pixel array that's only about 8mm (about 1/3") long. But despite its much smaller physical size, the new sensor actually has more pixels than the old one! Each pixel is just much smaller, obviously. To bridge the gap between the 3" plunger travel and the 8mm sensor window, we have to add something the old design didn't need: optics. We need a lens, to focus an image of the plunger onto the tiny sensor, just like in a conventional camera.

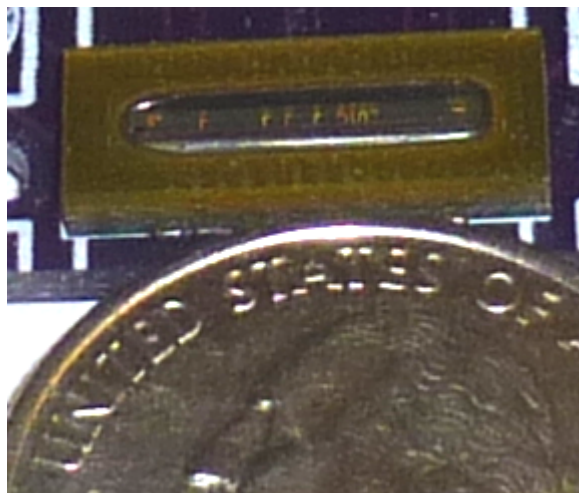
The need for a lens makes the system a little more complex than the old one, which is unfortunate. But the lens has a major upside, too, which is that a focused image gives us better accuracy in determining the plunger position. The old "shadow" image was inherently fuzzy, so it didn't take full advantage of the native pixel resolution of the sensor. With a lens involved, we can capture a much sharper image, which lets us determine the plunger position with greater precision. The new sensor has higher native resolution than the old one to start with, so when we add in focusing optics, we get a system with significantly better performance than the old one (which was already very good).

Introducing the TCD1103 sensor

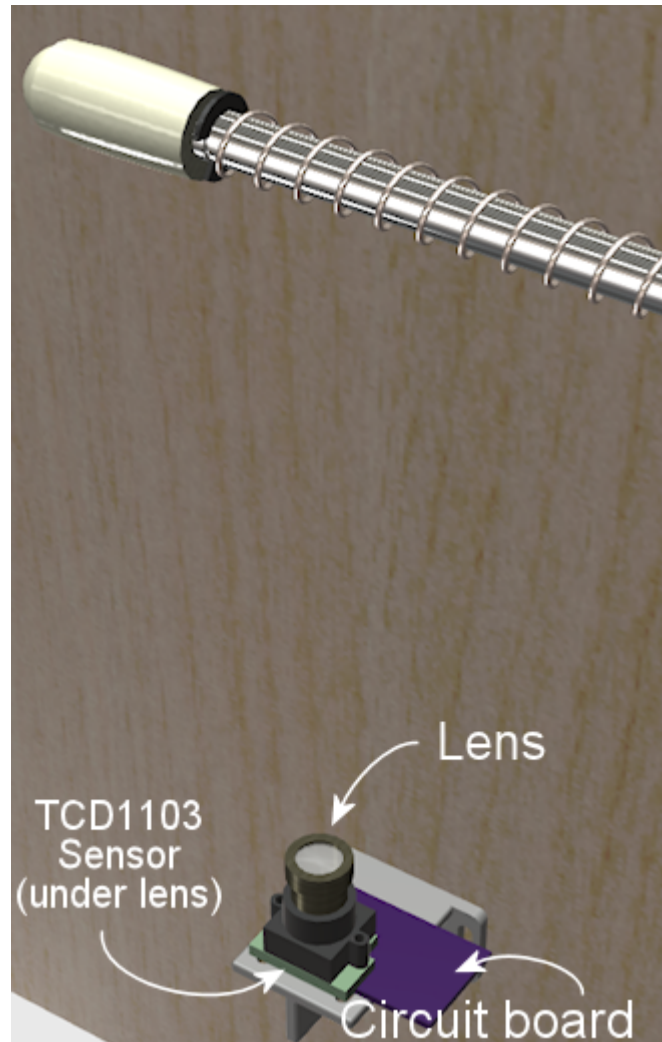
The Toshiba TCD1103 is an optical linear image sensor chip. This is similar to the image sensor in a digital camera, but rather than having a rectangular array of pixels like in a camera, the TCD1103 has all of its pixels arranged into a single row. That's the "linear" part. To be more specific, the TCD1103 has 1500 pixels in a row about 8mm long.



With a US quarter, for scale:



This sensor can be made into a limited sort of optical camera by fitting it with a lens to focus an image onto the pixel file. We can use this to track the the plunger position by pointing the camera at the plunger, with the pixel file oriented along the plunger's axis of motion. The camera will see an image of the plunger, and as the plunger moves back and forth, we'll see the image move along the pixel file. With a little image analysis, the software can figure out where the tip of the plunger is along the pixel file, which in turn tells us how far back the plunger is being pulled.



To keep the image analysis simple, we'll set up the plunger so that it creates a certain image pattern on the sensor. We'll put a bright white wrapper on the tip of the plunger, so that the end of the plunger stands out as a clear bright area in the image, and we'll place a dark (matte black) background behind the plunger, to create high contrast between the plunger tip and the background. This makes it easy for the software to find the end of the plunger in the image: it just looks for the edge between the dark background and the bright spot at the plunger tip.

This is probably my favorite plunger sensor out of all of the current options. It can read the plunger position to a very high degree of precision, thanks to the 1500-pixel resolution of the TCD1103 and the focusing optics; the plunger position can be reliably determined to better than one part in a thousand over its range of travel, which works out to around 1/400 of an inch. The sensor is fast, capturing images about every 4ms, or 250 times a second, which allows the software to accurately track the high-speed motion when you pull back and release the plunger. And it has the inherent "non-contact" advantage of an imaging system: there's no mechanical

contact between the plunger and the sensor, so there are no moving parts in the sensor to wear out.

The only disadvantage of this sensor system is that it's a bit complex to build, in that it requires the lens, a mounting apparatus, and some extra electronics to interface the sensor to the KL25Z. The plans presented below include parts lists, 3D-printable designs for the mounting bracket, and a custom circuit board layout with all of the interface electronics, so you won't have to improvise any of the elements yourself. Most of it is just a matter of gathering the necessary parts from various sources, and we'll give you details on exactly what to buy and where to look for it. But there is one special task required that does add a degree of difficulty, which is that the circuit board uses surface-mount (SMD) parts. That can be a little intimidating if you haven't done SMD soldering before. If you're willing to give it a shot, you might find that it's not actually not that hard, and we include detailed instructions to try to make it as approachable as possible. But it is a task that many people find intimidating, so I wanted you to be aware of what you're getting yourself into.

Bill of materials

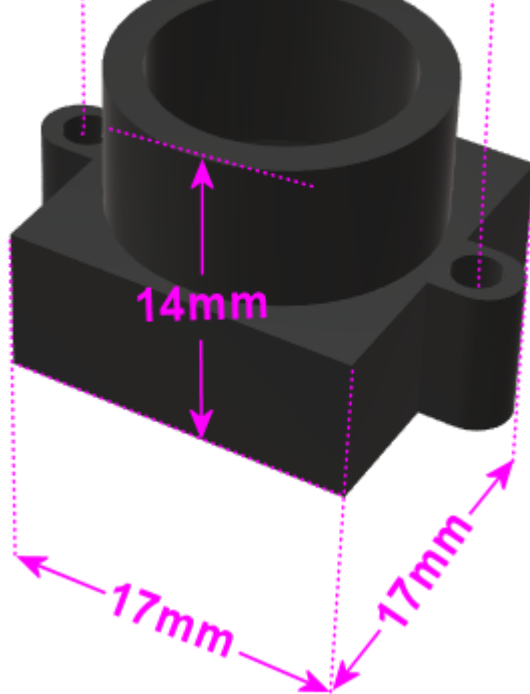
This project requires a fairly large set of parts. A few have to be custom-made, and the rest are off-the-shelf items, mostly things that are easy to find. Here's the full list:

- Standard pinball plunger assembly; see "Plunger" in Chapter 20, Cabinet Parts List for part numbers
- M12 lens, 16mm focal length, 1/2.7" image format. These can be found on Amazon and other online vendors for under \$10. Make sure to match our specs for focal length and image format. Some lenses will list compatibility with multiple image formats; that's fine as long as 1/2.7" is included among the compatible formats.

Note that the figure **1/2.7"** is the image format that the lens is designed for, which corresponds to the sensor size. It looks confusingly similar to the "f/" figure that's always quoted with lenses for real cameras, but it's a completely different thing. (The "f/" number is a measure of the lens aperture size, which determines its light-gathering power. It's a dimensionless ratio.) Many of the vendors selling M12 lenses seem to be confused about this, so you might see weird figures like f/2.7" or f/16mm, which are of course bogus. I've also seen "megapixels" figures like 3MP or 5MP, which are meaningless with a lens. If you find a candidate lens whose seller seems too confused to quote the proper specs, you might want to look elsewhere rather than guessing at what the seller really meant.

- M12 lens holder with 20mm mounting-screw spacing and an overall height of 14mm, as depicted in the diagram below. Most of the ones I've seen for sale match these size specs, but there are some other sizes available, so check the product dimensions against the diagram. The mounting screw spacing is critical, to fit our circuit board and mounting bracket designs, and the height is important because it affects the range of adjustment for focusing the lens. Suitable lens mounts are available from Amazon and other online vendors for about \$5 for a pack of 10. (The lens mounting can't be 3D-printed, because the threading patterns for the lens and mounting screws are too fine.)





- M12 lock ring, to help keep the lens from moving once you've set the focus properly. M12 lenses usually come with these, but you can also find them as separate parts if needed, on Amazon and other sites. Look for "M12 lock ring".
- 3D-printed mounting bracket. No special materials are needed; any type of plastic print material should produce acceptable results. The design can be printed on a home 3D printer, or by an online service such as All3DP or Shapeways. Ordering it through an online service should run under \$10. The plans are available here:

`TCD1103_bracket.stl`

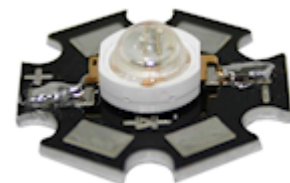
- Custom circuit board. You can have this fabricated by OSH Park, Elecrow, or other PCB makers; we explain the procedure below. The custom board at OSH Park costs about \$6.50. The EAGLE plans are here:

`tcd1103-interface-board-smd.zip`

- PCB stencil. This is another custom part that you can fabricate from the EAGLE plans for the board, for about \$8. This is optional but recommended, since it makes the SMD soldering work much easier. We describe the procedure for ordering a stencil below.
- The electronic components for the circuit board. See the TCD1103 section in Chapter 91, Electronic Parts List.

Important: **don't** break the seal on the TCD1103 sensor chip's anti-static bag until you're ready to solder. Don't even open it long enough to peek at the chip. The chip is sensitive to ambient humidity, so the packaging was prepared at the factory under super-dry conditions and sealed against moisture. It's important to keep the packaging sealed until you're ready to install the chip. See the "Warning on humidity" below.

- 1W red LED, to serve as a light source. Suitable LEDs are readily available on Amazon and eBay. I like the "star" LED type depicted at right. These are fairly easy to solder, and the base serves as a heat sink and is easy to attach to a mounting bracket. If you happen to have extra extra 3W RGB star LEDs of the



type often used in Chapter 56, flasher panels, you can use one of those, with just its red channel powered.

- A 10Ω, 2W resistor for the LED
- A 1" metal "L" bracket, to serve as a mounting bracket for the red LED. You can find these at any hardware store or home improvement store.
- Thermally conductive paste or tape (also known as "heat sink" paste or tape), for attaching the LED to the "L" bracket
- M2-0.4 x 12mm (also known as 2mm x 0.4mm x 12mm) machine screws, quantity 2, for attaching the lens mount to the bracket. Note that the lens mount I'm using takes **M2** metric screws, but some other types might use **#2** screws instead (the non-metric sizing system, which is *almost* but not quite the same size). #2 x 1/2" should fit if you need the non-metric type. Check the product page for the lens mount you buy, but I've never seen anyone bother to say which type is needed, so you might have to end up guessing. I'd just buy both screw types (M2x12mm *and* #2x1/2") to be sure you have the right one on hand when it comes time to assemble everything.
- #6 x 1/2" wood screws, quantity 3, to fasten the sensor bracket and light source to the cabinet wall

Tools for SMD soldering

This circuit board uses all SMD parts, most of which have small, closely-spaced pin pads - especially the star of the show, the TCD1103 sensor chip. SMD chips are difficult to solder the old-fashioned way (that is, using a soldering iron to heat each pin and melt solder onto it individually). In fact, regular soldering is practically impossible in this case because of the physical design of the TCD1103, which has all of its pins fully concealed *under the chip*, with no exposed metal that you can hit with the soldering iron. Soldering this type of part requires a whole different technique.

But the "whole different technique" is actually *easier* than regular soldering, given the right tools. The industry didn't migrate to SMD parts to make assembly harder, after all! Granted, the main motivation was that SMD assembly is easier for robots, but it's also faster and easier for hand-assembly if you have the right tools.

The right tools consist of:

- SMD soldering paste. Solder paste is the secret sauce that makes SMD assembly easy. When cold, it's a tacky paste (as you might guess from the name). It lets you stick the parts to their assigned places on the board, and holds them there until the heating step. When heated, it melts into a combination of liquid solder and "flux", which helps confine the solder to the metal pads and keeps the solder from forming shorts to adjacent pads. You just heat up the whole board to solder-melting temperatures, and all of the parts are simultaneously soldered to their proper places. It's almost magic. (This is also what makes it so much easier than regular soldering: you don't have to laboriously heat each connection point individually. You just heat the whole board at once.)

A popular choice that I've seen many people recommend is Chip Quik SMD291SNL50T3, available in 50g tubs for about \$17 from Amazon, SparkFun, and other online sites. That's lead-free, so it's somewhat easier on your household air quality than leaded solders. I've also had great luck with MG Chemicals 4860P-35G, although that comes in a tube that's better suited for small jobs with a few pads.

- A solder stencil for the board we're building. This is a plastic or metal sheet with holes etched to match all of the SMD pin pads on the board. It (obviously) has to be custom-made for the board. A stencil isn't absolutely required, as you *can* apply the solder paste by hand to the individual pads. But it's quite difficult to get the right amount of solder in the right places with a complex board like this one, especially given how tiny many of the pads are. A stencil makes it easy: you position the stencil over the board, align the holes with the pads, smear a glob of solder paste over the whole stencil, and wipe off the excess. This leaves the paste deposited wherever there was a hole in the stencil, which should exactly line up with the pads on the board. You can have the stencil made at OSH Park's sister site, OSH Stencils, for about \$8. It increases the overall project cost, but it makes the assembly process so easy and reliable that I think it's well worth it.
- A heat gun. You can also use a toaster oven or a skillet on a stovetop, but for a small board like this, I prefer a heat gun, since it's easy to control and easy to see what you're doing. Specialized heat guns for SMD soldering are available, but they're expensive, and I've found that a cheap hardware-store heat gun works just fine for a small board like this one. I have a heat gun from Harbor Freight Tools that cost about \$20, which I've successfully used to solder many small SMD boards.
- Tweezers or forceps. Optional, but very helpful for manipulating small parts. I found a pair of anti-static tweezers on Amazon (Vetus Pro ESD-safe fine-tip curved tweezers, model ESD-15, \$2-\$10 from various Web sellers) that have worked well for me.
- PCB jigs. Optional; you can also just use some spare blank circuit boards if you have them. These are just some plastic pieces the same thickness as the circuit boards (typically 1.6mm) to place around the board to keep in place while you're working with the stencil. It's the thickness that counts, which is why spare blank circuit boards work as improvised jigs; the idea is to make a flush surface around the edges of the boards with something of the same thickness. If you order the stencil from OSH Stencils, they'll throw in a jig set for \$6.
- Solder paste spatula/spreader. Professional metal and plastic spreaders are available at professional prices, but most hobbyists just use something improvised, like a putty knife from a hardware store. Old expired credit cards are supposed to work pretty well, too. OSH Stencils throws in a credit-card sized plastic spreader as a freebie with each order.

Order the circuit board

You can have the circuit board custom-made at OSH Park, Elecrow, or any other circuit board maker.

The EAGLE plans for the board are available here:

[tcd1103-interface-board-smd.zip](#)

Since this board is small, I'd highly recommend OSH Park if you're in the US. They're inexpensive, fast, top quality, and their ordering process is delightfully easy:

- Open the OSH Park site in your browser
- Upload the **.brd** file from the EAGLE plans linked above
- Click through to the ordering page and place the order

OSH Park will send you three copies of the board for about \$6.50.

Ordering from other vendors: If you prefer to order the boards from Elecrow or another PCB vendor, the process is a little more complex. Other vendors don't usually accept EAGLE .brd files like OSH Park does. Instead, they require you to upload a format known as "Gerber" files. The reason we don't just include the Gerbers in the ZIP is that they have to be generated separately for every PCB maker, using a "design rules" file provided by your chosen manufacturer. You use EAGLE to generate the Gerbers once you have the design rules file from your PCB vendor. It's easier than it sounds, but requires a couple of extra steps on your part. The full procedure is explained in Chapter 93, Fabricating the Expansion Boards. Follow the instructions there, using the .sch and .brd files from the ZIP download linked above. That chapter also has tips on selecting a PCB vendor, and instructions on how to place an order with the typical vendors. You can use the same PCB manufacturing options recommended in that chapter for the expansion boards - the same options will work equally well with the sensor board.

Order an SMD stencil

As mentioned earlier, I highly recommend using a stencil to aid in the SMD soldering for this project. This circuit board has a lot of small and closely-spaced pads; it's difficult to get the solder paste in the right place by hand. A stencil adds to the cost, but it makes the solder paste placement easy and reliable. You're much more likely to be successful on the first try with this board if you use a stencil.

You can order a stencil from OSH Park's sister site, OSH Stencils. Or, if you're having your boards made by Elecrow or another PCB maker, you can have them make the stencil along with the board order. (Note that you can always mix and match vendors, too. OSH Stencils might be cheaper for the stencil even if you order the board from Elecrow, so I'd check pricing at both sites before finalizing your order.)

The ordering process at OSH Stencils is a little more complex than OSH Park's, but it's not too difficult:

- Go to OSH Stencils in your browser
- Click **Upload design** and select the .brd file from the ZIP linked above. You might be prompted to create an account or log in, so that the upload can be stored for later access.
- In the uploaded file list, you should see three entries, with suffixes **.gbp** (bottom stencil), **.gtp** (top stencil), and **.gko** (board outline).
- The bottom stencil (**.gbp**) for this project is empty, so you should simply delete it from the list by clicking the big red X at the right
- Make sure that the other two are set to their proper types in the drop lists: **.gtp** should be **top stencil** and **.gko** should be **board outline**. The site usually gets these right automatically, but check, and change the drop list settings if they don't already match.
- In the **Project** name box at the top left, enter a name for the project, and click Create Project
- In the Project Name column of the file list, set each project to the new project name
- Click **Generate Stencils** at the top right
- This will take you to a Projects list view. Click on the project name in the list to open it, then click on the Top Stencil item to preview it. The preview layout

should look like the arrangement of solder pads on the board.

- In the Stencil Settings, select your preferred options for the stencil. You can read about the various options in the site's help/FAQ pages, but here's what I'd recommend:
 - Material type: Polyimide film, 5mil
 - Border size: 0.75

If you plan to mass-produce the board, you'll want to upgrade the material to stainless steel. It's durable enough to make probably hundreds of copies, but it'll add about \$5 to the cost. Polyimide is a plastic film that's cheaper but less durable; it's what I'd choose if you only plan to make a few copies.

- Click **Add this Stencil to Cart** at the top right
- Go to the cart and check out to complete the order

Ordering stencils from other PCB vendors: If you're ordering the PCBs through Elecrow or another vendor, they usually give you the option to also make the SMD stencils as part of the order. That's a "checkbox" option with most of the other PCB vendors - you just check a box on the order form saying you want to include the stencils, and they automatically generate the stencils from the same Gerber files you prepared for the PCBs. There will probably be some other options that you have to choose, such as stencil material, thickness, border size, and whether or not a "frame" is included. Since this is a small and relatively simple board, I'd just choose the cheapest options. In any case, you **don't** need a frame - frames are mostly for factory machines that apply the solder paste mechanically, and since those are industrial equipment with \$100K+ price tags, I'm going to guess you don't have one at home. (And if you do, you know way more about PCB stencils than I do!)

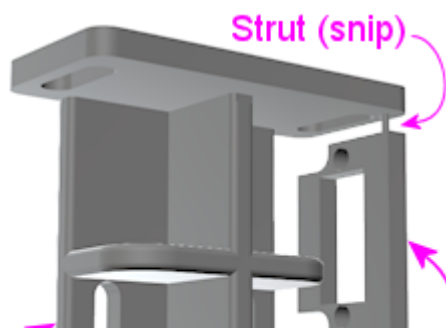
Fabricate the 3D-printed bracket

The plans for the mounting bracket can be downloaded here:

`TCD1103_bracket.stl`

No special materials are required for this print job. You can print this on a home printer, or use any online vendor. I've had great results with All3DP lately; they have a pricing engine that gets quotes from several vendors, which found good prices for the parts I've tried. I've also used Shapeways for several projects, and I've always had good results with them.

The printed part actually has two components, printed together into a single block using the "model airplane" approach. The main component is the bracket itself, and the smaller attached component is a spacer that goes between the circuit board and the lens mount, to position the lens mount properly above the sensor chip. To separate the two parts, use scissors or wire cutters to snip the little plastic strut connecting them. You can trim off and discard the strut.





Install the plunger

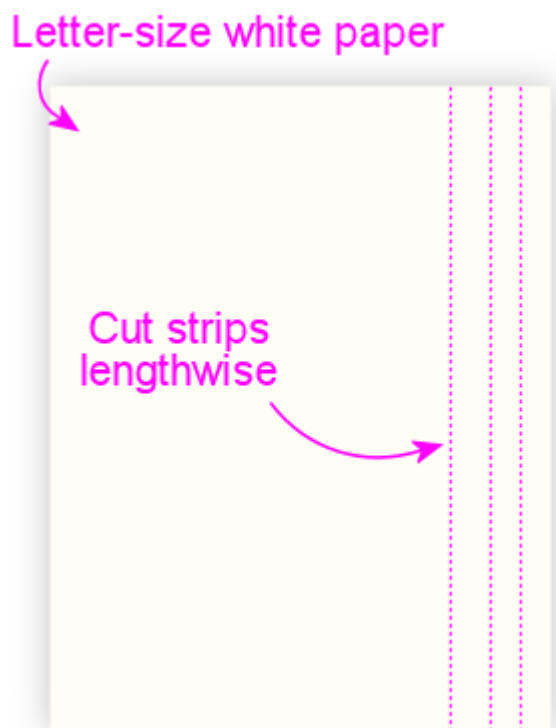
While you're waiting for the circuit board and 3D print job to arrive, you can get the plunger itself set up in your cab. The sensor doesn't have any mechanical connection to the plunger, so you don't have to wait for the rest of the parts to install the plunger. There's also nothing special you have to do to install it; just set it up like a regular pinball plunger. See "Plunger" in Chapter 23, Cabinet Hardware Installation for assembly instructions.

Set up a reflector on the plunger

We have to make one small modification to the standard plunger assembly. We're going to replace the standard rubber tip that's normally included in a real plunger with a reflective paper cylinder. This cylinder will serve as a reflector, which will show up as a clear, bright region in the image on the sensor.

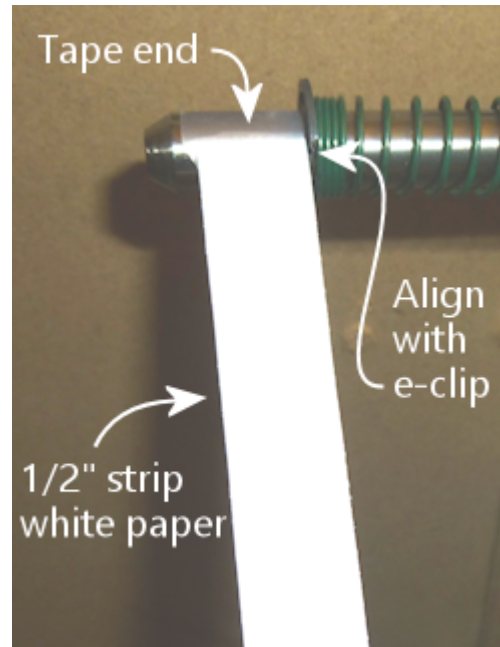
We use the paper reflector for a few reasons. One is that white paper is more reflective than the white rubber they use in the plunger tips, so we'll get a brighter image with a paper reflector. Another is that the rubber tips are rounded at the end, and we'll get a sharper edge in the image (which is what the software looks for) with a squared end. A third reason is that we can make the paper reflector a little bigger than the rubber tips, which will give us more leeway in aiming the lens, which makes installation easier.

- Remove (or skip installing) the rubber tip at the end of the plunger. If you bought complete ball shooter assembly with the rubber tip already installed, you can remove it simply by pulling it off the end; it's just held on by friction.
- Grab some ordinary white paper, letter size
- Cut three strips lengthwise: 1/2", 1/2", and 3/4"



$\frac{3}{4}$ " $\frac{1}{2}$ "

- Using cellophane tape or masking tape, tape one end of one of the $\frac{1}{2}$ " strips to the plunger tip, aligning one edge of the paper with the e-clip that holds the spring on.



- Tightly wrap the strip around the plunger to form a cylinder.



- Tape down the end of the strip
- Repeat with the second $\frac{1}{2}$ " strip
- Repeat with the $\frac{3}{4}$ " strip
- You should now have a $\frac{3}{4}$ "-long paper cylinder at the end, about the same diameter as the e-clip





- Check clearance with your flipper switches by pulling the plunger all the way back and making sure that the paper cylinder doesn't catch on the switches. If you don't have enough clearance, you can try wrapping the paper strip a little more tightly, or you can trim a little of the length off one of the strips and re-wrap it.

Q: "Why the different strip widths?" A: To make the outer edge sharper. We want the sensor image to see a sharp edge at the end of the plunger, so that the software can identify the boundary between the plunger and the dark background as clearly as possible. A clear, sharp edge in the image makes the position reading more accurate and stable. I noticed that if you make the whole strip the same width, some of the inner windings can poke out a bit beyond the end of the overall cylinder, which can make the edge a little softer in the image. Making the inner layers narrow makes them less likely to jut out.

Q: "Why isn't this a 3D-printed part?" A: It certainly could be! But I figured it would be better to save the extra expense given that it's easy enough to craft this out of paper. Plus, the point is to create a reflective surface that will show up well in the sensor images, and white paper is a better material for that.

Install a dark backdrop above the plunger

The next step in setting up our little photo studio is to place a dark backdrop behind the plunger. And by "behind", we really mean "above", because the camera is going to be positioned near the floor of the cab pointing up at the plunger.

The dark backdrop is there to ensure that the white reflector we just installed at the end of the plunger stands out with good contrast in the sensor images. That'll make it easier for the software to get a solid reading on the plunger position when analyzing the sensor images.

All you need to do is to install some matte black material just above the plunger. Be sure to use something with a matte finish, not anything shiny. Black felt is probably the ideal choice, but black construction paper also worked well in my tests.

I'll leave the details of how to install this up to you, since the most convenient arrangement will depend on how your cab is laid out. You'll almost certainly have something already positioned just above the plunger - either the underside of your TV, or the bottom of your "apron". Whatever is there, it should be enough to fasten black paper or black felt to the bottom of it. I'd try to cover the whole area around the plunger, from front to back and about an inch on either side.

If you have lights in your flipper buttons, you might also consider installing something to block that light, so that it doesn't hit the black backdrop. That probably isn't strictly necessary, but given that this is an optical sensor, the less stray light the better.

Warning on static electricity

As with many electronic parts, the TCD1103 is sensitive to static electricity. This chip is actually **very** sensitive to static, even more so than most other components, so please take extra care handling it. All of the normal procedures needed for any static-sensitive parts apply (see Chapter 67, Static Electricity Precautions), but I'd be extra vigilant with this part because of its high sensitivity.

Warning on humidity

As though static electricity weren't enough to worry about, the TCD1103 chip is also sensitive to humidity - at least, it is during the soldering process. After soldering the part, humidity won't affect it, so you don't have to worry about ongoing problems if you live in a humid climate. But humidity is a big issue for this type of chip during the soldering process, because of the high temperatures involved. Any moisture that leaches into the chip's plastic casing can form steam bubbles when the chip is heated for soldering, and these can vent explosively, cracking the plastic case and destroying the internal wiring and silicon parts.

When you buy this chip from a reputable vendor like Mouser or DigiKey, they'll ship it in a sealed, moisture-proof plastic bag, which keeps the chip dessicated during storage. **Don't open the bag until you're ready to solder the chip**, to ensure that it's not exposed to ambient humidity too early.

The packaging should include an indicator card (sealed inside the moisture-proof bag with the chip), with chemical indicator spots that turn different colors according to humidity exposure. When you first open the bag, immediately check the indicator card to make sure it shows the proper color (instructions will be printed on the card). The indicator card is there to ensure that the seal on the bag was intact throughout the storage period, so as long as it shows the proper color when you first open the bag, the chip is safe to solder.

(Note that the indicator card will start changing colors quickly after you open the bag, since your house is almost certainly more humid than the dessicated conditions required for storage. That's okay! The card isn't there to check your house's humidity; it's there to assure you that the package seal wasn't broken during storage. It's only important to check the indicator when you first open the package.)

Once the packaging is opened, the part will start absorbing ambient moisture from the air, but fairly slowly. It remains safe to solder for up to **120 hours** (five days) after opening the bag. This assumes that your ambient relative humidity level is 60% or below; if you live in an especially humid area, the safe period will be shorter. The best bet is always to keep the bag sealed until just before you're ready to assemble the board. And no need to panic or rush once you do; you'll certainly be safe for several hours no matter how humid your climate is.

I wouldn't count on moisture-proof handling and packaging if you bought from a cheaper source like eBay or Aliexpress. If there's no indicator card enclosed, you should use the baking procedure described below to be safe.

"Baking" procedure: If you have a chip that wasn't kept in moisture-proof conditions during shipping and storage, there's a fairly simple procedure called "baking" that you can use before soldering to make sure it's free of moisture. You should do this if any of these apply:

- Your chip didn't come with a moisture indicator card in the packaging
- The indicator card showed excessive moisture when you opened the package
- You opened the packaging but didn't complete soldering within 120 hours (less if you live in an especially humid climate)

The procedure for baking the chip is simple. You just need to pop it in an oven, toaster oven, or under a heat lamp, where you can maintain a temperature of 125° C (about 250° F) for 24 hours. After keeping it in a 250° F environment for 24 hours, take the chip out and let it cool.

The baking procedure will gently release any trapped moisture. After baking, the chip is once again safe for soldering, and should remain so for up to 120 hours, as though you had just taken it out of its original factory packaging.

Build the circuit board

Be sure to read the sections above about **static electricity** and **humidity** dangers. Please observe all of the precautions diligently - the TCD1103 can be easily destroyed by improper handling.

This board uses surface-mount parts, so it can be a little intimidating if you haven't done SMD soldering before, but it's not as hard as it might seem. If you're new to this, you might want to take a look at a tutorial or two to get an idea of what's involved and some tips on technique; try a Web search for "SMD stencil tutorial". I'd once again recommend using the stencil with this project because of the small parts involved.

The SMD stencil soldering process works like this:

- Set up your work area on a large flat surface, like a workbench or kitchen table
- Put down a fairly large piece of cardboard or heavy paper (butcher paper, for example)
- It's easiest to hold everything in place using a PCB jig set - acrylic corner pieces the same thickness as the board, that you can place around the board to keep it from sliding around and to form a uniform flat surface for a couple of inches around the board's perimeter. You can also use some spare boards for this, placing them around the perimeter of the board you're building.
- Tape down the board and the jig pieces to the work surface, using masking tape. Be sure not to cover up any of the SMD pads on the subject board with tape.
- Place the stencil over the board and line up the holes in the stencil with the solder pads on the board. Tape it down to the work surface (leaving all of the holes uncovered, of course).
- Scoop a big glob of solder paste onto the board on one side
- Use the spreader to smear it across the board to the other side to form a thick layer, pressing it down as you go to force it through the holes
- Now do another sweep with the spreader in the reverse direction, this time scraping off as much of the top layer as you can, so that the only paste remaining is the paste that squeezed through the holes. Return the excess paste to the tub.
- Carefully peel the stencil off of the board
- Clean any remaining paste off of the stencil and return it to the tub
- You should now have a nice clean layer of paste on each of the solder pads on the board. Check the alignment to make sure that the pads line up reasonably well. You don't have to get the alignment perfect! Solder paste is very forgiving; its affinity for the metal pads will make it ball up nicely on the pads when heated even if you got a fair amount outside the lines.

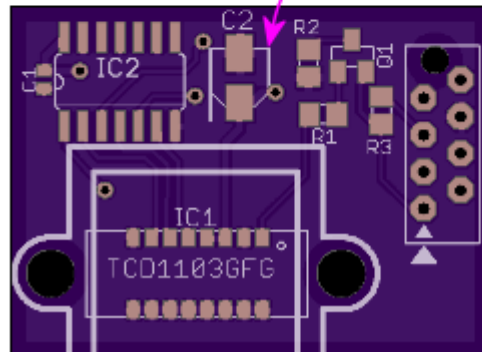
- Place the parts! Use tweezers for the smaller parts.
 - Match the parts to the pads on the board using the "designator" - R1, C1, etc
 - As you place the parts, press down on them gently with the tweezers to stick them to the paste
 - Resistors aren't polarized, so it doesn't matter which direction they go
 - Capacitor C1 isn't polarized, so its orientation doesn't matter
 - Capacitor C2 **is** polarized, so its orientation does matter. The part's plastic base has an asymmetrical shape that serves as the orientation guide. Match the shape to the outline printed on the board.

Capacitor C2 package
(overhead view)



Note asymmetrical
shape of base

Match shape
to outline
on board

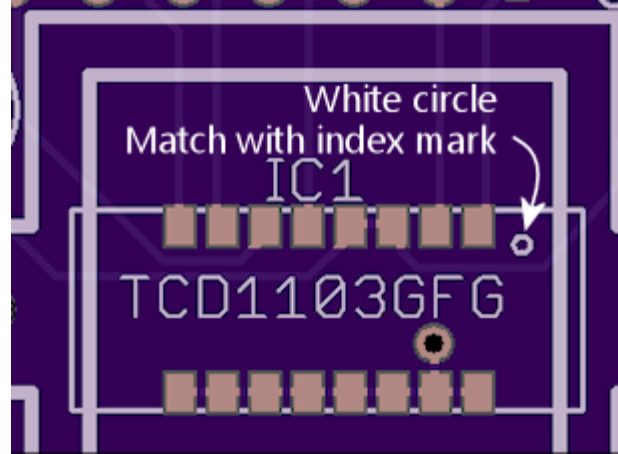


- The transistor's (Q1) orientation should be obvious from the pad layout
- Make sure that the sensor chip is oriented correctly: the white circle printed on the board in the corner of the corresponds to the "index" mark - a dot - printed on the **bottom** of the chip next to pin 1. Note that the index mark is printed on the bottom of the chip, so you'll have to do a little spatial reasoning to keep track of where it goes when you flip the chip over.

Index mark



Bottom of chip



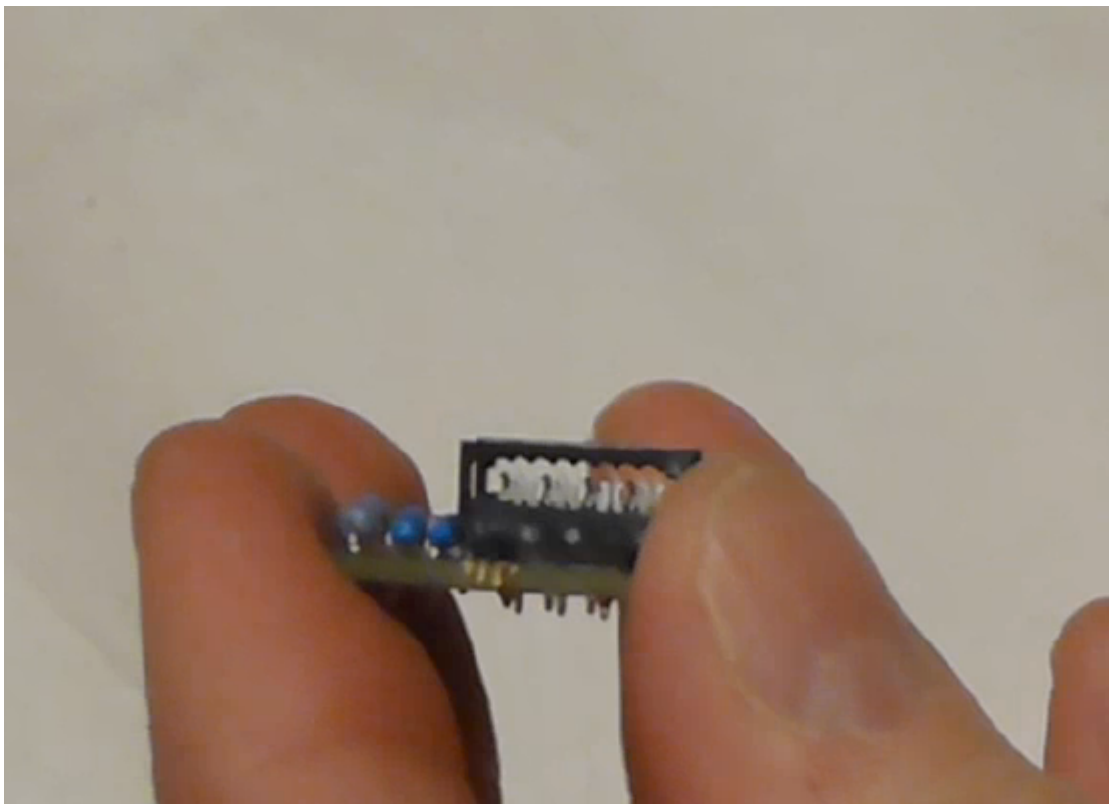
- The other IC chip (IC2) also needs to be oriented properly, in this case by matching the little half-moon notch on the chip to the notch printed on the circuit board outline
- Once all the parts are placed, it's time to solder. The basic pattern for SMD soldering is to heat the whole board **gradually** for about 60 to 120 seconds, so that nothing cracks or pops from thermal expansion, then keep it hot enough to melt the solder for about 30 to 60 seconds. The natural surface tension of the melted solder will make it flow over the pins and pads, so the goal is simply to melt the solder across the whole board for long enough for this flow process to happen. Then we can let the board cool, again going gradually to avoid thermal shock. Everything should freeze into place as the solder sets. I prefer the heat gun method:
 - Remove the board from the work surface and put it on something heat-safe
 - Set the heat gun to low
 - Point the gun at the board and hold it about an inch from the board, straight above
 - Make small circles over the board to heat the board evenly
 - Do this for about 60 seconds. This is the ramp-up phase - we're trying to heat the board gradually here so that there's no thermal shock.
 - Set the gun to high, continuing to point it at the board and make small circles
 - Watch the solder paste; in 30-60 seconds, it should start melting; it'll change from the dull gray paste color to bright silver
 - Once all of the solder is melted, continue heating for about 10 or 20 seconds more to let the solder flow over the pins and pads
 - Now gradually back the gun away from the board - two inches for a few seconds, three inches for a few seconds, etc. Keep going like this for about 30 seconds.
 - Turn off the gun
 - Let the board cool
- Once the board has cooled, there's still one part that has to be soldered the old-fashioned way, namely the ribbon cable connector. Insert that through the holes and solder the pins on the bottom side of the board. Be sure to protect the TCD1103 while the board is flipped over, by putting down some cardboard or paper on the work surface.

You can skip the ribbon cable connector if you plan to use hook-up wire instead of a ribbon cable. I'd recommend the ribbon cable with this board, not only because it's easier (in my opinion), but because it makes a cleaner signal path for the high-speed data signals this sensor uses.

Install the ribbon cable

Remember to continue being careful about static electricity while handling the board. The CCD chip remains sensitive to static even after soldering. (Humidity is no longer a worry after soldering, though.)

If you're using the ribbon cable connector, the next step is to attach the cable. You should already have soldered the connector to the board (see above). The connector comes in two pieces: a base with the pins sticking out, and a clip that fits over the top. The base is the part that you solder to the board. If you separated the top clip part during soldering, put it back, but only loosely - you want to leave a gap between the two parts big enough for the cable. If the clip is still in place, loosen it to open up a gap for the cable.



Slip the ribbon cable into the opening. If your cable has a stripe down one side (usually red), oriented the cable so that the stripe is on the side with the little white triangle printed on the circuit board. The triangle printed on the board and the stripe on the cable both represent "pin 1" in the wiring.

If your cable **doesn't** have a stripe, you should add one with a red marker. It's best to use a permanent marker with **oil-based ink** for this, because water-based inks tend to rub off of the plastic insulation too easily. Draw a stripe down the whole length of the cable along one side. (It doesn't matter which side you pick.)

The cable should just barely fit into the gap in the connector. This is part of the design, to make sure the cable is positioned properly side-to-side. Let the end of the cable extend about 1/4" past the clip.

Once the cable is positioned properly, get out some pliers. There's actually a

specialized IDC crimper tool for this job, but ordinary pliers work well enough if you're careful. Carefully apply pressure to the top of the clip. Start at one side of the clip, push it down just a little bit there, then gradually move to the other side. Work back and forth a few times until the clip is all the way down and snaps into the locks. It's better to do this gradually so that you force the wire down fairly evenly and keep it close to level side-to-side.

Be careful not to dislodge the soldered pins on the bottom of the board! If you have a spare copy of the board, you can put the spare board under the assembled board, with the connector pins going through the holes in the spare board. This will let you apply pressure to the bottom of the board without applying pressure to the bottom of the pins.

Finally, install the IDC connector at the far end of the cable. This is the plug that connects to the expansion board header. This is almost exactly like attaching the soldered connector.

Be sure to line up pin #1 on the plug with pin #1 on the cable. On the plug, you should find a small triangle or arrow at one corner, embossed in the plastic. (It can be hard to spot since it's just a little bump, but it should be there.) That's the pin #1 side. Make sure that lines up with the red stripe on the cable. When you plug the cable into the expansion board, this side should also line up with the triangle printed on the expansion board next to the header, which is yet another pin #1 marking.

Connect and test

Before installing the sensor in the cabinet, it's a good idea to connect it to the KL25Z and test it out, to make sure the hardware is working correctly. This will also let you use the Config Tool's plunger sensor viewer to view the live images coming in on the sensor, which you'll need to be able to do during the installation process to aim and focus the lens.

Before connecting the sensor, it's a good idea completely unplug the KL25Z to remove power. You should unplug the KL25Z USB cables even if the PC is powered off, because most PCs continue to power USB devices even when the main PC power is off.

Continue to be careful about static electricity when working with the sensor board. The CCD chip is highly static-sensitive even after soldering.

Connect to the expansion boards

If you're using the Pinscape expansion boards, simply plug the ribbon cable connector into the main expansion board's "Plunger" header. Be sure to line up the "pin 1" triangle printed on the expansion board with the red stripe on the cable.

Connect to a standalone KL25Z

If you're using a standalone KL25Z, I'd still recommend using the ribbon cable connector. That provides a cleaner signal path than ordinary hook-up wire would, which is important for this sensor because it uses high-speed data signals. The easiest way to connect the ribbon cable to a standalone KL25Z is using the Chapter 100, plunger sensor breakout board:

- Plug the ribbon cable into one header on the breakout board (it doesn't matter which one), lining up the red stripe on the cable with the "pin 1" triangle printed on the board
- Connect the pins on the other header on the breakout board as follows:

- Breakout board **3.3V** to KL25Z P3V3 (pin 8 on J9)
- Breakout board **GND** to KL25Z GND (pin 12 or 14 on J9)
- Breakout board **D0** to KL25Z PTE21 (pin 3 on J10)
- Breakout board **B0** to KL25Z PTB0 (pin 2 on J10)
- Breakout board **E20** to KL25Z PTE20 (pin 1 on J10)
- Breakout board **D5** to KL25Z PTE22 (pin 5 on J10)

If you're already using some of those GPIO pins for other purposes, you can use different pins for the sensor connections, with a few constraints. First, the wire that we connected to PTE21 requires a PWM-capable pin. Second, the wire we connected to PTB0 requires an analog-input (ADC) pin. The other two connections (the ones assigned to PTE20 and PTE22) don't have any special requirements, so they can be reassigned to any free GPIO pins. The Config Tool will show you the allowable GPIO ports for each connection when you configure the sensor in the Settings page, and you can also check Appendix 1, KL25Z Pin Out for the full list of GPIO pin capabilities.

Configure the software

- You should have the sensor connected to the KL25Z at this point, as explained above
- Connect the KL25Z to the PC using USB cables as normal
- Bring up the Pinscape Config Tool on the PC
- Go to the Settings page
- Scroll down to the plunger setup section
- In the Plunger Type drop list, select **TCD1103**
- If you're using a standalone KL25Z, check that the GPIO pin assignments match your wiring. If you didn't use the default pin assignments, click on the pin name boxes and update the pin assignments to match the pins you actually wired. Note that the "fM (Master Clock)" and "OS (output signal)" connections have special requirements that limit the choice of GPIO pins; the Config Tool will show you the valid pins for those. If you wired those connections to GPIO pins that aren't allowed, you'll have to change wiring to valid pins as shown in the Config Tool.
- If you're using the expansion boards, there's no need to select the pin assignments, since they're pre-determined in the expansion board wiring
- At the bottom of the page, click **Program KL25Z** to save the changes to the KL25Z firmware

Initial testing

We won't be able to get sharp images on the sensor until we install the lens, which we'll come to shortly, but we can at least do some rough initial testing to see if the sensor is responding to light at all. Basic light reception is good enough to verify that the soldering process was successful and that the software is configured correctly. I find it very helpful to do these tests before attempting the full installation, so that I don't have to wonder if the basic wiring is working while trying to align the lens.

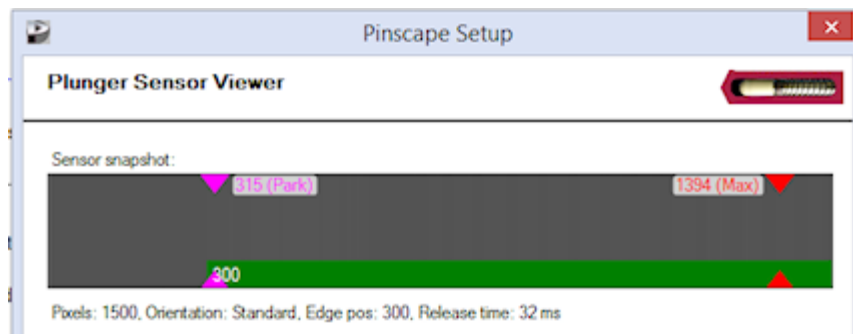
- Return to the main page in the Config Tool
- Click on the Plunger icon to bring up the plunger viewer

Note: During testing, **don't** point any light sources directly at the sensor.

Overexposing a CCD can cause a cascade effect that maxes out all of the pixels and gets them stuck like that until you reset power. Ambient light in a normally lit room should be enough to see a response on the sensor.

The plunger viewer shows the image being received on the sensor. Remember that this sensor only has a single row of pixels, so you shouldn't expect to see a full rectangular image like on a normal camera. Instead, the image is just a single line of pixels. The viewer screen expands this into a wide bar, but at each point along the bar (left to right), you're only seeing a single pixel value.

Without a lens installed, the image is so unfocused that it can really only show the ambient light level. The CCD sensor is extremely sensitive to light (especially at the red end of the spectrum), so if the sensor is sitting out in the open and you have the lights on in the room, the whole viewer should look fairly uniform, from dark gray to light gray, depending on how bright the ambient light in the room is. (Don't pay any attention to the green bar; without a lens to form a focused image, the software can't get a good read on the plunger position, so the green bar will probably jump around randomly.)



Typical snapshot with the sensor working. The pixels appear with a fairly uniform shade of gray. This particular snapshot has the sensor in a shaded area, so it's not picking up much light. With more ambient light, the pixels will be lighter gray or closer to white, but should still look fairly uniform. If the pixels look "noisy", with a lot of variation from one pixel to the next and a lot of rapid fluctuation, there might be a problem in the wiring or pin assignments.

The most basic test you can do is to completely cover the sensor with something opaque. You should see the sensor view change to darker gray, still fairly uniform across the image. Uncovering the sensor should return to a lighter gray or white image. It's okay if it's not completely black when covered and not completely white when uncovered. The important thing is to see a clear response to different light levels - brighter pixels when the sensor is exposed to light, darker pixels when it's shaded.

Another test you can try is to cover a portion of the sensor, using an opaque piece of thick paper or cardboard. If you cover a portion of the sensor window (holding the paper right up against the sensor), you should see a corresponding section of the pixels in the viewer window darken. Without a lens, you'll only be able to see a very coarse and blurry image, but it should still be possible to see that a section of the sensor is shaded.

If you see the expected responses to these tests, the sensor is working! You can proceed to the cabinet installation below. If the sensor doesn't respond as expected, the first troubleshooting step is to confirm that all of the GPIO pins for the sensor are assigned correctly in the Settings page. Carefully check each pin assignment against the physical wiring - the sensor won't work at all if even one pin assignment is wrong. (Incorrect pin assignments shouldn't cause any permanent damage to the

electronics, though.)

If the pin configuration looks right, you might try a power cycle reboot of the KL25Z. Overexposing the sensor beyond a certain point can cause a cascade effect that blinds all of the pixels, which can last until you power cycle the chip. That's something you can do by accident when you're handling the board during testing, so it can be worth trying a power cycle reset.

If the pin assignments are all correct and power cycling doesn't help, you probably have a problem in the soldering. Check all of the solder joints carefully (I'd use a magnifying glass). If you spot any solder joints that don't look properly connected, you can try using a fine-tipped soldering iron to heat the joint long enough to melt the solder. That's often all you need to do to fix a bad SMD joint - the natural surface tension in the melted solder will usually "heal" a bad joint.

What it looks like if it's *not* working

The typical appearance if the sensor isn't wired correctly is a "noisy" image - pixels rapidly fluctuating with different shades of gray, and random variation from one pixel to the next. The KL25Z reads pixels from the sensor through a GPIO analog input pin, so if the sensor isn't working, the analog pin will typically read random electrical noise, which shows up in the visualization as random and rapidly changing pixel brightness. When the CCD is working, the pixel readings tend to be stable over time and vary smoothly across nearby pixels.

If the pattern on the image viewer isn't "noisy", but doesn't respond as expected to changing light levels or to a partially covered sensor window, the sensor might actually be working; the problem might simply be that your ambient lighting is too strong or too weak. In other words, you might be overexposing or underexposing the sensor. If the sensor viewer is showing a uniform dark gray or black, try increasing the ambient light level in the room, and repeat the basic tests; if it's showing uniform bright gray or white, try decreasing the ambient light level.

Wire the light source

For the light source, I recommend a 1W red LED of the "star base" type. You can find these on eBay and Amazon.

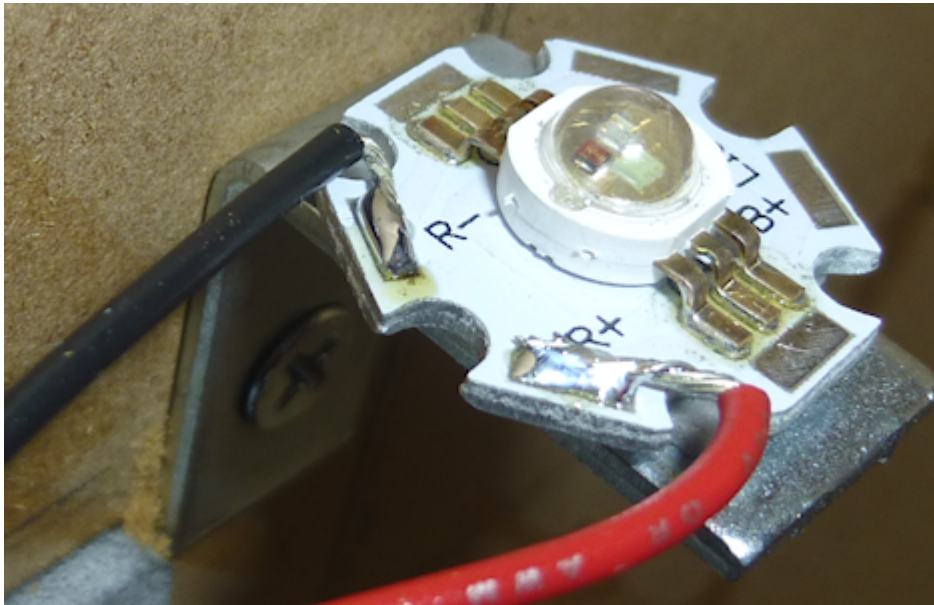
A 3W RGB LED will work equally well, because that's a combination of 1W red, blue, and green LEDs in a single housing. You can just wire up the red segment and leave the blue and green segments unused, which will make the light the same as a 1W red LED. Note that it might seem like powering the other segments could only be better in terms of providing even more illumination, but I'd actually recommend against that because we don't want to overexpose the images. 1W worth of red light seems to be just about the perfect amount in my testing.

The reason we use a red LED as the light source is that the TCD1103's peak sensitivity is at the red end of the visible spectrum. The sensor actually responds to light across the whole spectrum, so it's not absolutely required that we use a red light source; it's just that it's most sensitive to red light, so a red light source is the most efficient. We might need more power with a blue light to get a good exposure level.

- Figure how long a wire you'll need to reach from the light source, which will be installed at roughly the front right corner of the cabinet, to a connection point for your 5V power supply
- Cut a length of red hook-up wire and a length of black wire long enough to

reach

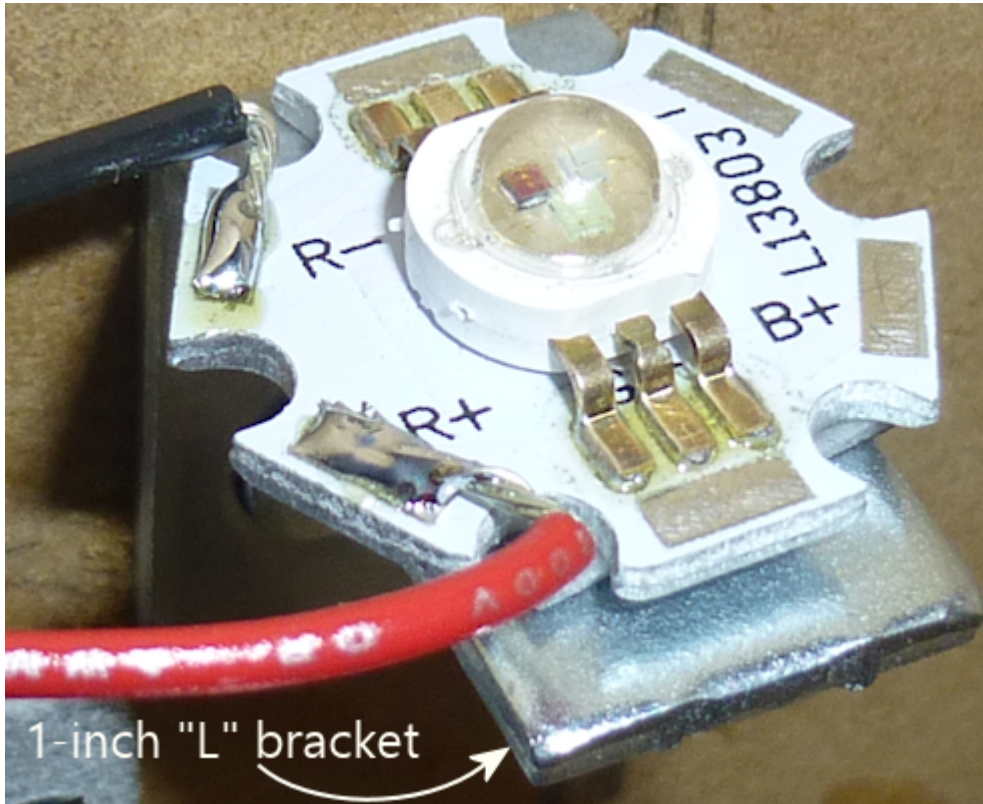
- Strip about 1/4" of insulation from each end of each wire
- Solder one end of the red wire to the "+" pad of the star LED (or the "R+" or "RED+" pad, if you're using an RGB LED). As always, the best soldering technique is to hold the wire and pad firmly together, press the soldering iron tip against the junction, wait a few seconds for the parts to heat, and then touch the solder to the **parts** (**not** the soldering iron tip) until the solder melts and flows freely over both the wire and pad. Then remove the soldering iron tip and let the joint cool for about 10 seconds while holding everything perfectly still. For joining loose parts like this, I find it's a lot easier if you use clamps or paperweights to hold everything in place while you're working.



- Solder one end of the black wire to the "-" pad of the star LED (or the "B-" or "BLACK-" pad, if it's an RGB LED).
- Cut the red wire at some point along its length, perhaps 10" or so from the LED end
- Strip about 1/2" of insulation from the two new ends of the wire
- Solder the new wire ends to the leads of a 10Ω, 2W resistor
- Wrap electrical tape or heat-shrink tubing around the exposed resistor leads, to protect against accidental short-circuits with other exposed metal parts in the cab. (**Don't** cover the body of the resistor - that could make it overheat.)



- Attach the LED to a 1" "L" bracket, using glue, paste, or double-sided adhesive tape. I recommend a thermally conductive "heat sink" tape, since that'll make the bracket into part of the heat sink for the LED. Assuming you're using the "star" type, the aluminum star base is already a good heat sink by itself, but it doesn't hurt to add more cooling surface area given that we're going to leave this LED running continuously.



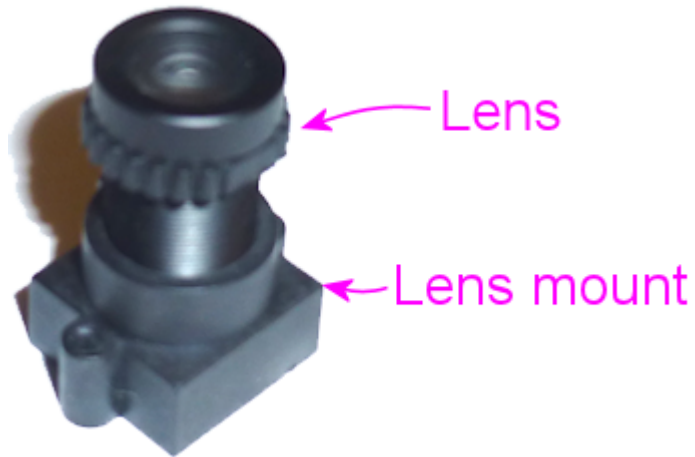
Assemble the sensor

- Fit a lock ring onto the lens, and screw it all the way to the top of (the wide part of the lens)

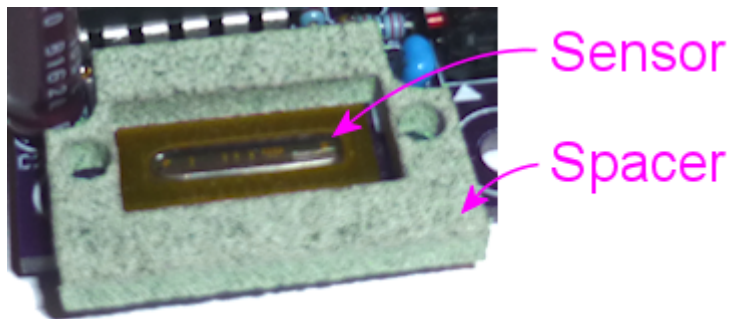




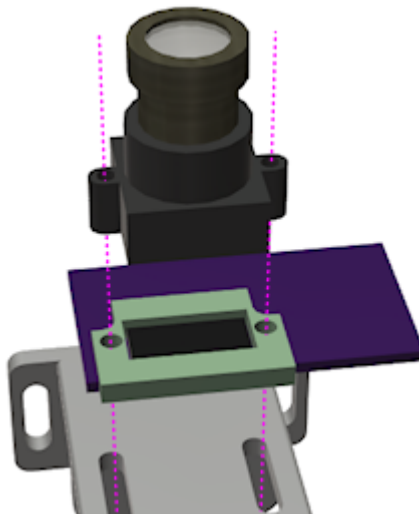
- Screw the lens into the lens mount

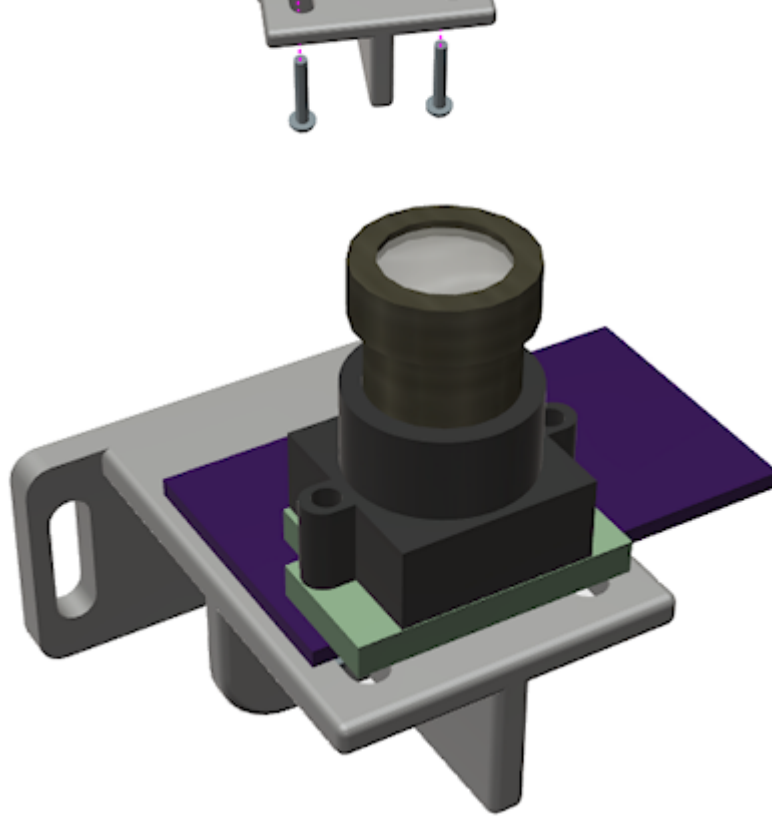


- Fit the sensor/lens spacer from the 3D-printed bracket over the sensor chip



- Assemble the mounting bracket, circuit board (with spacer), and lens/mount combo as shown below. Use M2 or #2 screws (whichever type fits your lens mount) to fasten the parts together. Tighten the screws just enough to hold the lens in position *loosely* - tight enough that there's enough friction to hold its position, but still loose enough that you can move it back and forth by hand. We'll need to be able to fine-tune the alignment later when we install it in the cab, so you want it to still be mobile at this point.





Install the sensor

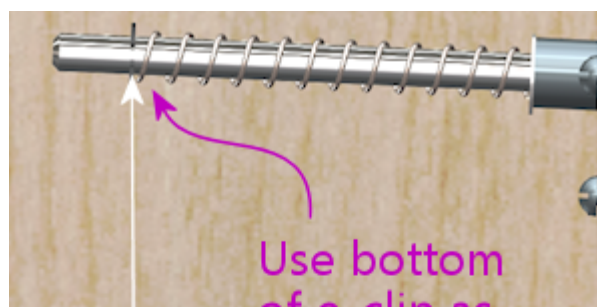
The sensor has to be carefully positioned to satisfy three constraints:

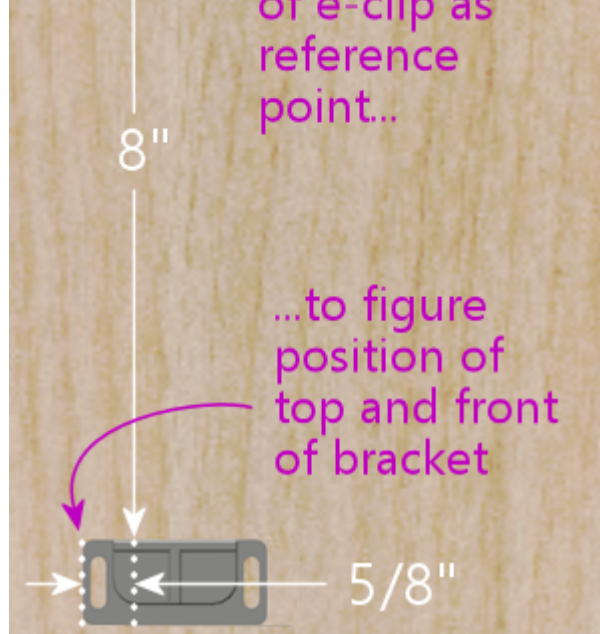
- It has to be a certain distance from the plunger vertically, so that the range of the plunger's travel will just fill the lens's field of view
- It has to be at a certain point along the plunger's travel front-to-back, so that the image can capture both ends of the plunger's travel range
- It has to be positioned side-to-side so that the lens points straight at the plunger

The mounting bracket has slotted holes to give you a little wiggle room to fine-tune the position and aim, but it's still important to get the overall initial position right.

Warning! Be prepared for a little bit of trial and error to find the ideal position. Given that we're using cheap no-brand lenses, I'm not sure how consistent the optics will be from sample to sample. There might be enough variation in the field of view that you might have to move the bracket a little bit up or down from what worked for me.

Here's the positioning that worked for me. This is based on the 3D-printed mounting bracket plans earlier in this chapter. Obviously, it's the position of the sensor that matters, not the position of the bracket per se. But the sensor position relative to the bracket is predictable, as long as we're all using the same bracket design, so it's easier to work in terms of positioning the bracket.





- Use the **bottom of the plunger e-clip** as the starting point
- Measure **8" straight down** along the cabinet wall, and mark this as the position of the **top of the bracket**
- From this point, measure **5/8" towards the rear of the cabinet**, and mark this as the **left side of the bracket**
- Now just fit the bracket so that its top left corner (as viewed in the diagram above) fits the corner spot we just marked

Don't worry too much about getting the measurements exact. The mounting bracket has slotted holes that will let you fine-tune the final positioning based on what you see in the sensor viewer, plus I tried to leave a little margin of error in these measurements so that it'll still work if you're a little off. Besides, as I warned earlier, your lens optics might be a little different from mine, so my measurements might not be quite right for your setup anyway.

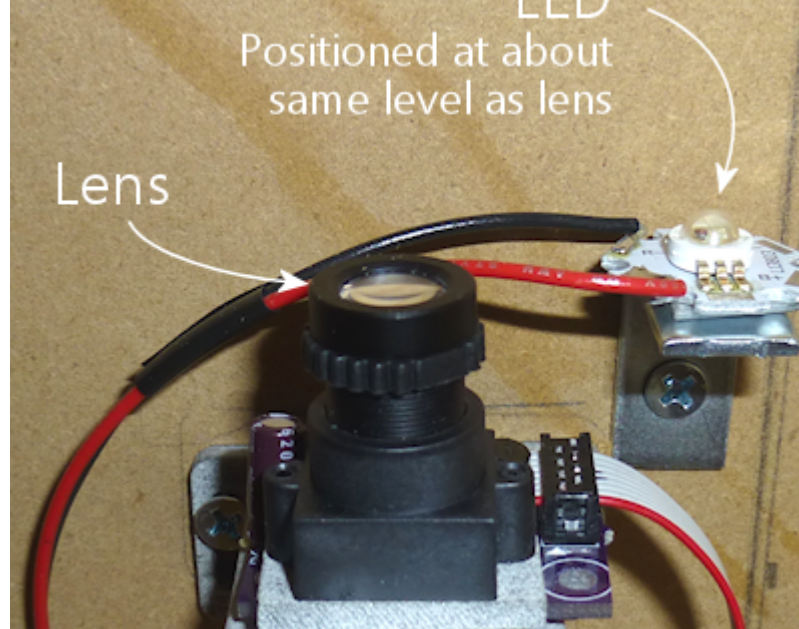
- Use the sensor bracket as a template to figure the screw hole positions
- Try to center the screws in the slots so that you have maximum room to adjust in both directions
- Drill 3/32" diameter pilot holes (about 1/4" deep) at the marked positions
- Install the bracket using two #6 x 1/2" wood screws
- Leave the screws a little loose for now so that you can adjust the angle of the bracket when we test it in the image viewer below

Install the light source

The light source is the next (and final) piece to install. You should already have wired the LED and fastened it to an "L" bracket for mounting in the cab. Now you just need to find a convenient place for it and install it.

I recommend placing the light source close to the sensor, at about the same level vertically as the lens. If you have room, I'd put it on the side of the sensor closer to the front of the cab, but I think it'll work fine on the other side, since these LEDs are very bright and not very directional.





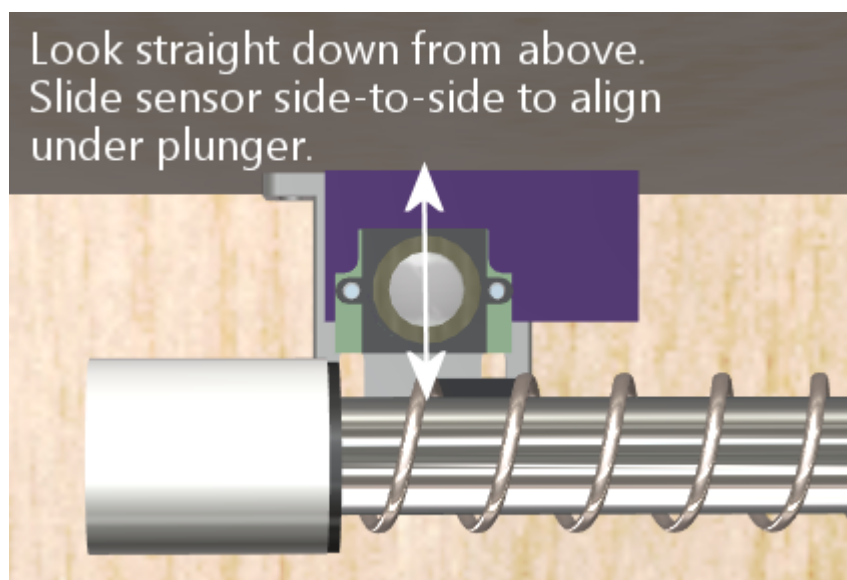
Install the LED's "L" bracket using a #6 x 1/2" wood screw. I like to drill a pilot hole (3/32" diameter, about 1/4" deep) first for this type of screw.

Aim and focus

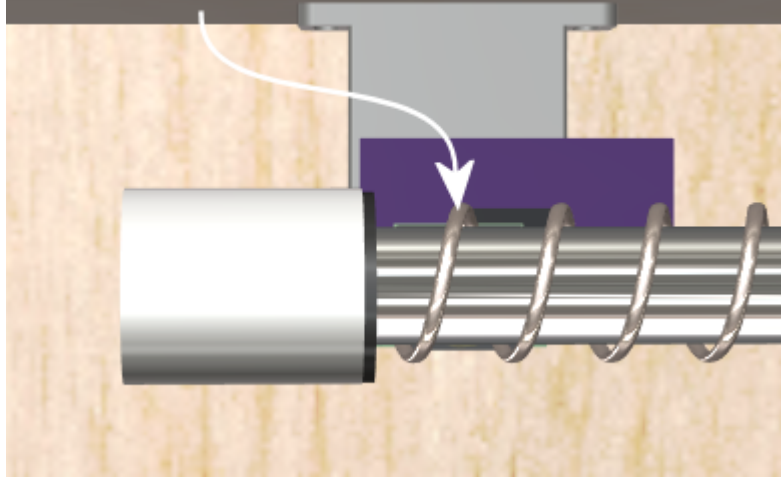
Now we have to align the sensor so that it gets a clear view of the plunger through the lens, across the plunger's whole travel range.

There are two parts to this. The first is to position the sensor side-to-side so that it's aligned with the plunger axis. The second is to adjust it front-to-back so that the plunger tip stays in view across its whole travel range. For that second step, we should only have adjust the angle of the bracket, so you shouldn't have to move the screws fastening it to the wall - there should be enough range in the slots that we can rotate the bracket to get the right alignment.

You can start the side-to-side alignment by visual inspection. Looking straight down at the plunger, slide the sensor side-to-side along the bracket until the lens looks to be centered under the plunger.



Lens is aligned under plunger

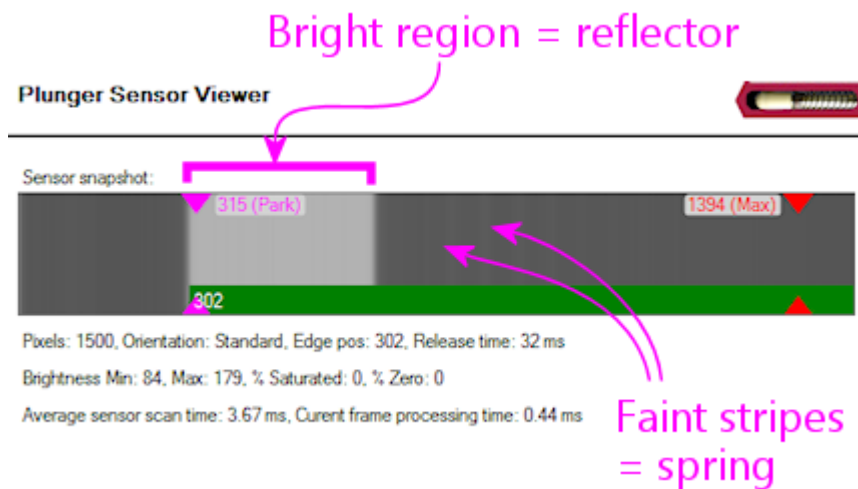


How precise do we have to be? The sensor's field of view side-to-side is only one pixel wide, which translates to a fraction of a millimeter along the plunger. The "target" that we're trying to image - the white paper reflector - is about a centimeter wide. So we only have to get our little one-pixel sliver to overlap that centimeter-wide target. It's a broadside-of-the-barn situation.

Even so, aligning it by eye is just a first step, to get us into the right neighborhood. To find the final position, we have to see what the sensor sees. Luckily, the Config Tool lets us do just that.

- If your dark backdrop isn't currently installed, put some kind of temporary dark backdrop in place - a piece of dark paper that you can lay on top of the cabinet above the plunger should be adequate.
- Turn on the LED light source
- Bring up the plunger viewer in the Config Tool (run the Config Tool and click the "plunger" icon on the main page)

What we're looking for is a clear bright-colored region that fills about a quarter of the field. That's the paper reflector we installed at the tip of the plunger. You might also see some faint alternating dark and light stripes; those are images of the spring and the plunger shaft.

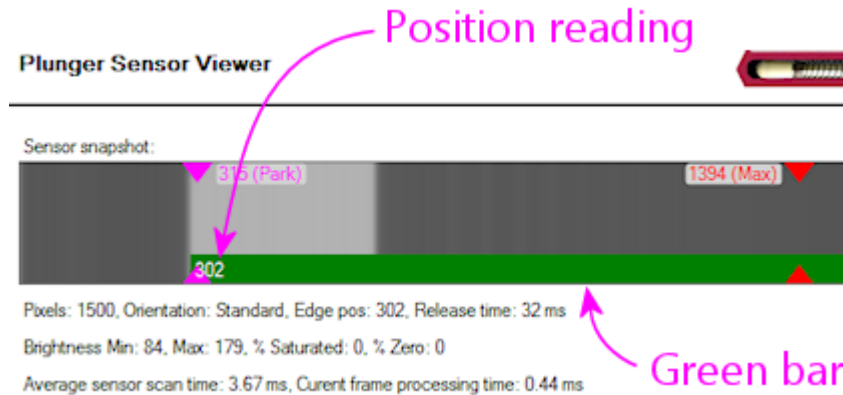


If you don't see the bright region corresponding to the reflector, try sliding the sensor side-to-side on the bracket. You might have to hunt around a little bit before the plunger comes into view.

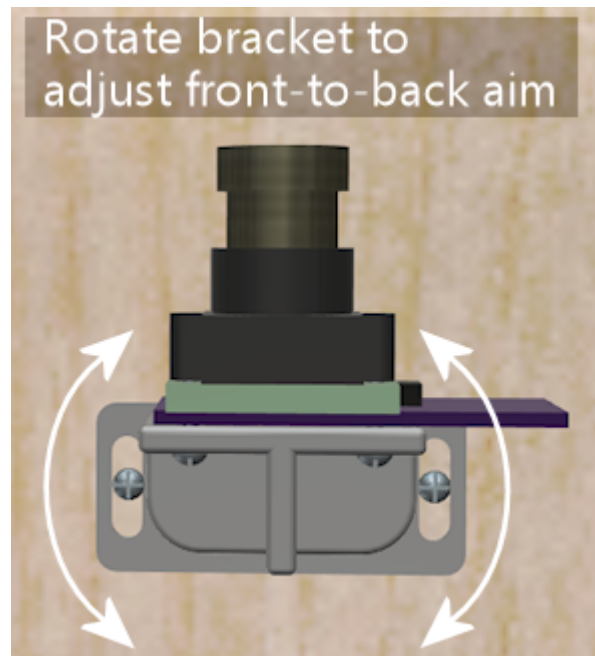
When you find the reflector, you should then fine-tune the position to try to get it as

close to the center-line as possible. That will give you the clearest image, and it should also help better tolerate vibration and small shifts in alignment. The plunger shaft usually has a little bit of play that lets it move around by a couple of millimeters in normal action, plus you have to expect all of the parts to move around a little bit over time.

Front-to-back positioning: Now that you have the reflector in view, the Pinscape firmware should be locking onto the plunger position. You should see this as a green bar showing the current position reading.



The goal now is to get the position reading (with the plunger at rest) to be at about 300. Do this by **twisting** the mounting bracket to re-aim the lens front-to-back. The slots in the mounting bracket are there to allow a little rotation like this.



Check the full travel range: Check that the leading edge of the reflector stays in view across the whole travel range:

- Pull the plunger all the way back
- Also push it forward, compressing the barrel spring as far as you can

It's only important for the **leading edge** to stay in view, since that's what the software looks for to find the position. It's fine if part of the reflector goes past the end of the window when the plunger is pulled all the way back.

If either end is out of view, try adjusting the mounting bracket rotation again to get it into view.

Things to watch out for:

- What if you simply can't get both ends of the range in view, because the field of view isn't wide enough for the travel range? This means that the sensor is too close to the plunger. The positioning I recommended is based on testing with my M12 lens with 16mm focal length, but I'm not sure how consistent these cheap no-brand lenses are from batch to batch. It's possible that the field of view won't be quite the same with different lenses, which might mean you have to adjust the distance between the sensor and the plunger. If the field of view isn't wide enough, you'll have to increase the distance.
- What if the bright region starts disappearing halfway along the travel range? This probably means that the sensor's pixel window isn't close enough to parallel to the plunger, so that the plunger moves "diagonally", relative to the sensor, as you pull it back. In this case, you'll have to try to get the sensor lined up better with the plunger axis, by rotating the sensor slightly on the mounting bracket.

Focus: We're almost done. The next step is to focus the lens. Use the viewer window to observe the leading edge, at the left side of the view. Adjust the lens to get the sharpest edge you can.





Finalize: Lock everything down in the calibrated position. Tighten the screws that hold the sensor to the bracket and the screws that hold the bracket to the cab wall. Set the focus lock ring, by screwing it down the lens until it locks against the lens holder.

Keep an eye on the plunger viewer throughout, to make sure that you don't knock things too far out of alignment while tightening any of the fasteners. I always find that things move around at least a little bit any time I adjust a screw, so do just a little bit at a time, monitor the effects in the viewer, and re-adjust the lens alignment as needed.

Calibrate: There's one last step, which is to run the calibration procedure in the Config Tool. This lets the software determine the resting position of the plunger on the sensor and measure its range of motion, so that it can report the position in terms that pinball simulators like Visual Pinball can understand. The simulators need the numbers to be expressed on an abstract scale where "zero" is the resting position, so the calibration procedure is needed to figure the translation between the raw pixel positions on the sensor and the abstract scale that the simulators use.

The calibration procedure can be accessed from the Config Tool's plunger viewer screen. Simply click the **Calibrate** button and follow the on-screen instructions.

106. Plunger Setup (VL6180X Distance Sensor)

I should start by saying that I don't consider this to be a very good option. I'm including it because it's easy to set up and fairly cheap, so it might be appealing to some people on those grounds. It also has the virtue of being a non-contact sensor, so it won't suffer any wear and tear from use. But its accuracy and speed are too low for me to recommend it. It's only accurate to about 1 centimeter, which translates to chunky animation of the on-screen plunger (the best available Pinscape plunger sensors are accurate to better than 1/100 of a centimeter). It's also comparatively slow, which makes it less responsive than other options. With those warnings in mind...

The VL6180X is a distance measuring sensor that works by sending out pulses of IR light aimed at a target, detecting the reflected pulses with a photosensor, and measuring the round-trip travel time of each pulse. It uses the travel time to determine the distance, using the known speed of light in air.

To use this for a virtual pinball plunger sensor, the basic idea is install the sensor at a fixed position just beyond the end of the plunger, with the sensor pointing at the tip of the plunger. As you pull back the plunger, the distance between the sensor and the plunger increases. The sensor's distance measurement tells us the plunger's current position.

VL6180X vs. VL53L0X

The VL53L0X is a newer chip from the same manufacturer with similar distance measurement features. It's not an outright replacement for the VL6180X, as it's not software-compatible and doesn't have exactly the same features. Its main advantage over the older chip is that it has a much higher maximum ranging distance, up to about 200cm vs only 10cm for the VL6180X. However, there's a tradeoff: it's considerably slower than the VL6180X. The VL6180X completes a distance measurement in about 13ms, whereas the VL53L0X requires at least 19ms, and is happiest at about 35ms. In my testing, the newer chip doesn't have any accuracy improvements over the old one.

Given the tradeoffs, I don't see any reason to use the VL53L0X over the VL6180X for plunger sensing. The longer range of the newer chip doesn't make any difference for us since we don't even need the full range of the older chip, and the reduced speed is a negative. If the newer chip's accuracy were noticeably better, that would make it more of a contest, but both chips seem to have about the same (poor) accuracy.

Electronics

You can buy pre-assembled VL6180X circuit boards from several robotics hobby companies. This is lucky for us, because the bare chips are rather challenging to work with. The pre-fab boards make them easy to use.

Here are three options:

- Pololu (\$12)
- Adafruit (\$14)
- Sparkfun (\$25, also sold at Mouser.com)

All of these options are pretty much equivalent. They all provide easy-to-solder terminals for all connections, and they all have the necessary on-board voltage regulators and level shifters to interface the VL6180X with the KL25Z. Any of these

boards can be used directly with the KL25Z hardware and Pinscape software; the only thing you have to do is run a few wires between the KL25Z and the sensor board.

The instructions below are designed around the Pololu board. I chose that one as the reference point because it's the cheapest and smallest option. You can substitute one of the others if you prefer, but be aware that you'll have to adapt the instructions below for any differences in the board you choose. The other boards are all very similar, so this shouldn't be difficult, but keep it mind as you build the project.

Wiring the board

The breakout boards all come fully assembled, except for external wire connections. The only thing we have to do is attach the wires between the board and the Pinscape controller.

Your board might come with a set of pin headers that you can optionally solder to the pads on the board. You can use these if you'd like, but I'd actually recommend soldering wires directly to the pin pads instead of using the pin headers. Using the pins creates a bit of extra work, since you'll have to build an extra connector to plug into the pins. I'd skip the pins and solder wires directly to the board terminals, since you really shouldn't ever need to disconnect the wires from the sensor board. We *will* use pluggable connectors at the other end, where we plug into the Pinscape board, so you'll still be able to easily unplug the whole assembly if you ever need to remove it from the cabinet.

I recommend 24 AWG stranded wire for the connections, but the exact gauge isn't important, as these wires carry extremely low power.

You'll need five wires, about three feet long each. Before cutting the wires, check the length you'll need by measuring the distance the wire will have to traverse to reach from your plunger housing to the Pinscape unit (the KL25Z or main expansion board). Take into account any extra length you need to route it around your PC motherboard or other obstructions.

Strip about 1/4" of insulation from each end of each wire.

Solder one wire to each of the following terminals on the board:

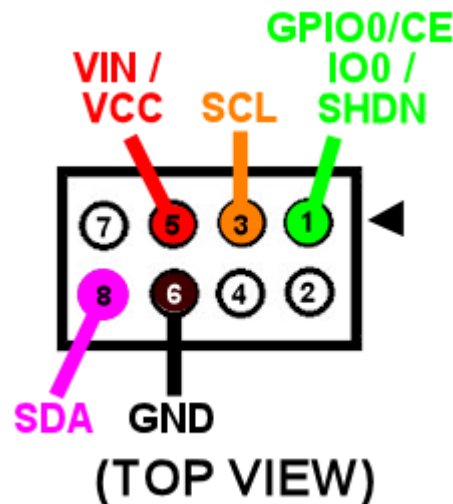
- VIN (Adafruit and Pololu) or VCC (Sparkfun) (**not** VDD or 2.8V)
- GND
- SDA
- SCL
- GPIO0/CE (Pololu), SHDN (Adafruit), or IO0 (Sparkfun)

Note that your board will have some other terminals besides those listed above. You can simply leave other terminals unwired.

Be careful about the "VIN" voltage input terminal! This one can be a little confusing because some of the boards have two "V" terminals. One is for the input voltage: this is the one we want to connect. The other is for the output of the on-board 2.8V voltage regulator. We **don't** want to connect anything to the 2.8V output. On the Adafruit board, the 2.8V output is plainly labeled as "2v8", while it's more obscurely labeled on the Pololu board as "VDD". In either case, just leave this terminal unconnected.

Expansion board wiring

For the expansion board connector, build a 4x2 crimp pin housing (housing, pins). First crimp a pin to the end of each wire (see Chapter 82, Crimp Pins). Insert the pins in the housing, following the diagram below for the pin placement.



It would be a good idea to put a mark of some kind on the housing in the corner next to pin 1 - the pin marked with the arrow on the diagram above. That will make it easier to remember which side aligns with the "pin 1" arrow marker on the circuit board when you plug in the connector.

Standalone KL25Z wiring

For the standalone KL25Z, I recommend using the Chapter 100, plunger sensor breakout board. Then you can just plug it into the breakout board.

- Build the expansion board connect as described above
- Build the ribbon cable connector exactly as described above, as though you were using the expansion boards
- Follow the instructions in Chapter 100, Plunger Sensor Breakout Board to build the breakout board
- Connect the following wires between the breakout board and the KL25Z:
 - Breakout board **3.3V** to KL25Z P3V3 (pin 8 on J9)
 - Breakout board **GND** to KL25Z GND (pin 12 or 14 on J9)
 - Breakout board **B0** to KL25Z PTB0 (pin 2 on JP10)
 - Breakout board **E20** to KL25Z PTE20 (pin 1 on JP10)
 - Breakout board **D0** to KL25Z PTE21 (pin 3 on JP10) (Yes, the label on the breakout board is different in this case)

If you prefer to use your own ad hoc wiring, see "Plug it in" below for the list of KL25Z GPIO connections.

Physical installation

I haven't come up with a reference design for the housing for this sensor, so you'll have to come up with something on your own. The basic idea is to mount the sensor at a fixed position in your cabinet so that it faces the tip of the plunger head-on. The point is to measure the distance between the sensor and the plunger.

There have been several commercial virtual pinball plunger kits available over the

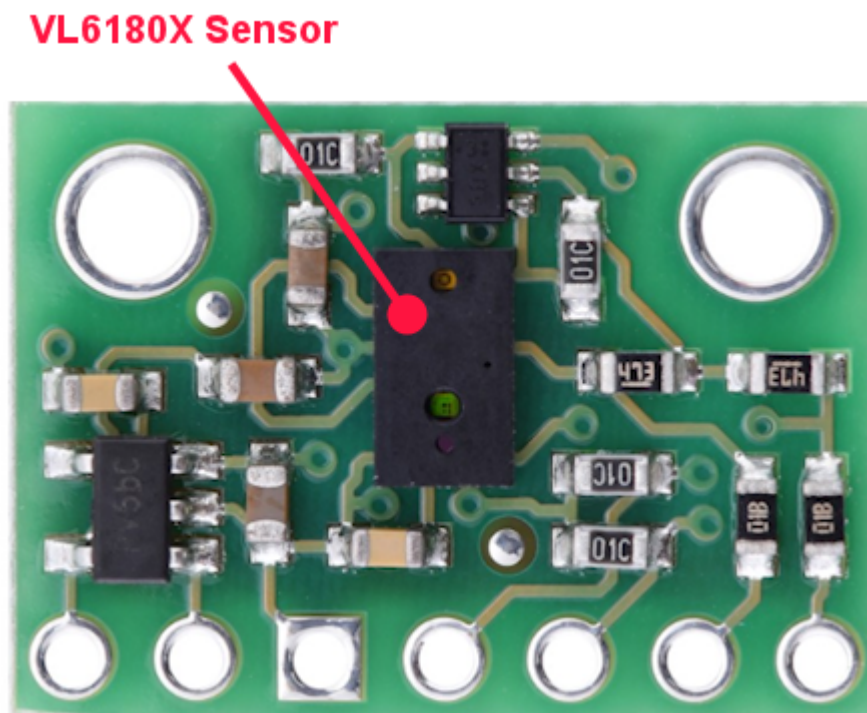
years that were also based on IR proximity sensors, so we can look to them as examples for how we might set this up. All of those used a plastic tube surrounding the plunger, with the sensor installed at the far end of the tube, facing the plunger tip. The plastic tube serves dual purposes: it provides a place to mount the sensor, and it shields the whole area from stray light that could interfere with the sensor's readings. The VL6180X actually doesn't actually need such shielding as much as the older sensors used in the commercial kits do, since it has a "laser ping" design that's pretty good at ignoring ambient light, but even so, it wouldn't hurt.

Choosing a material for the tube: This isn't as simple as it might seem! Intuitively, it seems like any kind of opaque material would be good. But the sensor uses infrared light (technically, 850 nm), and many materials that look opaque to our eyes are actually transparent to IR. The common 3D printer plastics PLA and ABS are both partially transparent to IR, so it would somewhat defeat the purpose to use them for a light shield. Black PETG is supposed to be nearly opaque to IR, so it might be a better choice. It might also be a good idea to make the inside surface texture of the tube or cover somewhat rough, to minimize reflections. A smooth surface that readily reflects the signal might be worse than no cover at all.

Choosing the sensor mounting position: Make sure that the sensor is far enough away from the end of the plunger's range of motion that the plunger won't ever hit the sensor. Remember that the plunger springs forward about half an inch from its resting position, because the barrel spring on the front can compress a bit. Leave a gap of at least 1 centimeter (about half an inch) if possible.

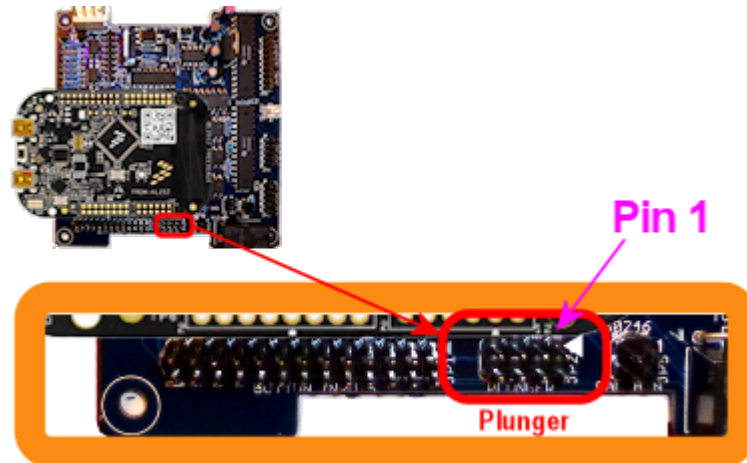
When you mount the board, be sure to mount it with the component side facing the plunger.

Here's what the Pololu board looks like. The sensor is the little black box in the middle with two small holes (one hole is the laser emitter, the other is the photosensor). The sensor chip looks the same on all of the boards, but the layout of the terminals and the other parts varies slightly.

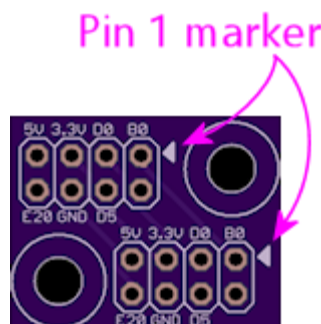


Plug it in

Expansion boards: Once you've built the connector as shown above, simply plug it into the plunger connector on the main expansion board. Make sure the plug orientation is correct by matching pin 1 in the housing (see the diagram) with the pin 1 triangle printed on the expansion board.



Standalone KL25Z: If you're using the plunger sensor breakout board (recommended), build the expansion board connector as described above, and just plug it in to the pin header on the breakout board. Be sure pin 1 on the plug (see the diagram) to pin 1 on the board, which is marked with a little white triangle printed next to the header.



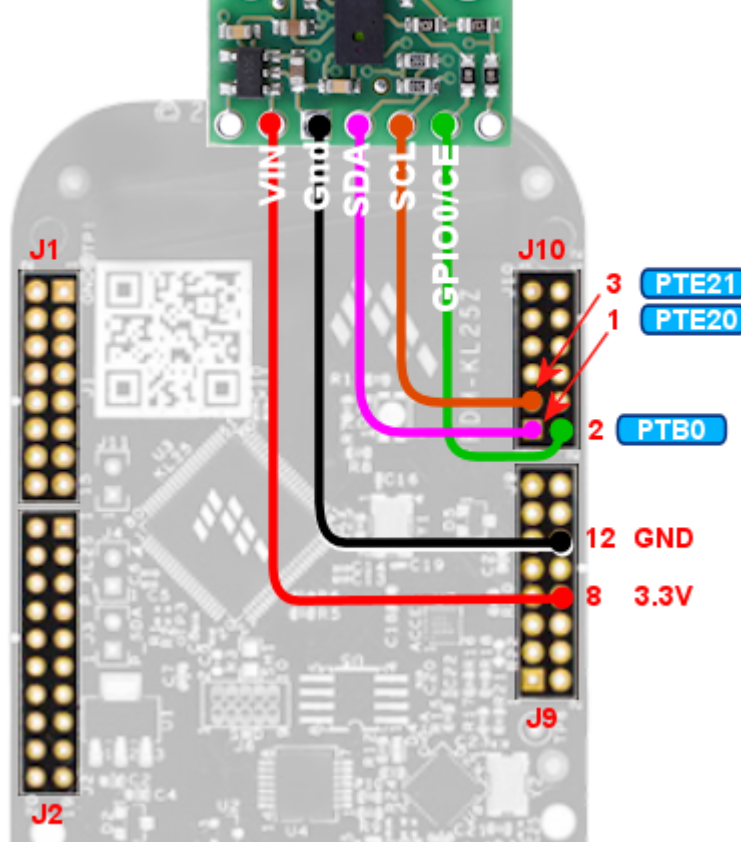
If you prefer to use your own ad hoc wiring, connect the wires between the board and the KL25Z as shown below.

Sensor Board Pin	KL25Z Pin
3.3V	P3V3 (JP9-8)
GND	GND (JP9-10)
SDA	PTE20 (JP10-1)
SCL	PTE21 (JP10-3)
GPIO0/CE	PTB0 (JP10-2)

Note that the last pin, GPIO0/CE, has different names on some of the boards: it might be labeled IO0 or SHDN. It's the same pin in any case; the different board makers just chose to give it different labels.

The illustration below shows the Pololu board. Be sure to adjust the pin ordering if you're using a different board. Just match the labels shown on the diagram to the labels printed on your board.





Note that the three GPIO ports listed above are only suggestions. If you're already using the same ports for other functions, you can assign the sensor inputs to other ports using the Config Tool. Any free GPIO ports can be used with this sensor (it doesn't have any special requirements for particular ports). The power and ground wires aren't configurable, though, so connect those as shown.

Software setup

Once you have the sensor physically installed and plugged in, run the Pinscape Config Tool on your PC. Go to the Settings page. (If you have multiple Pinscape units installed, choose the Settings page for the unit that's plugged into your new plunger sensor.)

Go to the Plunger Sensor section. Select VL6180X in the "sensor type" popup.

(If the VL6180X option isn't available in the plunger sensor list, you probably have an older version of the Config Tool. Updating to the latest version should add the option.)

If you're using the expansion boards, the pin settings will be set up automatically.

If you're using the standalone KL25Z, set the pin assignments for the three pins (SDA, SCL, and GPIO0/CE) to match the pins you connected on the KL25Z. The SDA and SCL pins should match the pins you wired to the like-named terminals on the sensor board. The last one, GPIO0/CE, goes by different names on the different boards: on the sensor board, it will be labeled as GPIO0/CE, IO0, or SHDN, depending on which type of board you have.

Save the new settings by clicking "Program KL25Z" at the bottom of the window.

You should now test and calibrate the plunger. Return to the home screen in the Config Tool and click the Plunger icon for the unit with the sensor attached. This will let you look at the raw sensor input. Move the plunger and make sure it seems to be

tracking properly.

If the sensor is working properly, click the Calibrate button in the plunger viewer window to begin the calibration process, and follow the on-screen instructions.

If the sensor doesn't seem to be working, go back to the Settings screen and double-check the sensor pin assignments. Make sure that none of the pins are marked with warning icons (⚠️). Check each wire and make sure that it goes to the proper pin on each end (KL25Z and sensor board). Check that each GPIO port assignment on the settings page matches up with the physical pin on the KL25Z and connects to the corresponding terminal on the sensor board.

Jitter filtering

The VL6180X generates distance readings in millimeters, but it's really only accurate to about a centimeter, so the millimeter numbers it comes up with are pretty much just wild guesses. As a result, the readings fluctuate quite a bit from one reading to the next, even when the target is standing still. With a stationary object, you'll see the readings jump around constantly within about 5mm of the true distance, plus or minus. This is annoying and unrealistic during virtual pinball play, because it makes the on-screen plunger dance around nervously when the real plunger is standing still.

The Pinscape firmware offers a "jitter filter" to deal with this. The jitter filter lets you set a range for ignoring random fluctuations. As long as the random noise from the sensor stays within the range, the device ignores the fluctuations and reports a stable, stationary plunger. The on-screen plunger only moves when the sensor readings move outside of the noise window.

To enable the jitter filter, run the Pinscape Config Tool and go to the Plunger Viewer window. There's a setting in this window for the jitter filter. To adjust it, start with the filter at zero, and gradually increase it until the green bar showing the filtered reading stops jumping around. Use the smallest value that gives you acceptable results, because larger values will reduce the usable precision during play.

Backwards operation

If the on-screen plunger moves backwards from the real plunger, you can fix it in the software without reinstalling the sensor. Open the Pinscape Config Tool. In the row for the controller, click the Plunger icon. Check the box for "Reverse orientation". (If it's already checked, un-check it.) This tells the software to reverse the readings from the sensor, so that it acts like it was installed in the opposite orientation.

107. Pinscape Plunger Calibration Button

The Pinscape firmware has a provision for a dedicated, hard-wired button to activate plunger calibration. This was a feature from the very first version, before the Config Tool existed. When the Config Tool came along, the dedicated button became unnecessary, since you can now run the calibration from the Windows UI. But the firmware still lets you connect a physical button for this if you really want to. I don't see any good reason to do so, as the whole process is much friendlier through the Windows UI, but I'll give you a quick guide to setting it up, just in case.

Parts

All you need is a pushbutton of some kind. The Chapter 91, Electronic Parts List part list includes the button I used for this in my cab, which is a small illuminated pushbutton that's similar in size and shape to the buttons in the Williams coin door service panel (E-switch part number WBL2UOABQR05CLR). But there's nothing special about that one other than the form factor; all you need functionally is a pushbutton. It's nice to use an illuminated button, since the firmware shows feedback on the progress of the calibration process by blinking the light, but it's not required.

If you're connecting an illuminated button to the standalone KL25Z, you'll also need a few electronic components (also listed in the parts list) to build a small transistor amplifier to drive the button lamp. The KL25Z on its own can't drive a lamp due to its low GPIO power limits. It's not necessary to build this extra piece with the expansion boards since it's already built in.

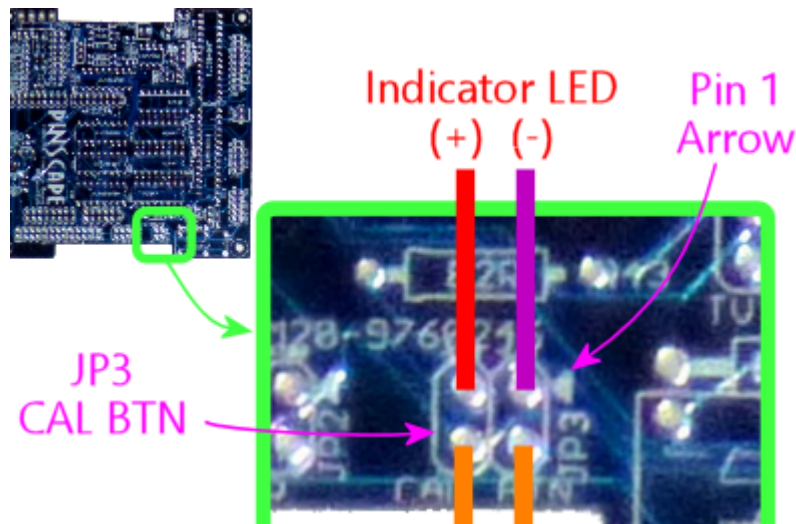
Where to mount the button

The calibration button is an "operator" feature, so I'd put it on the inside of the cabinet, making it accessible through the coin door.

My preferred location would be an added custom service panel mounted to the inside of the coin door, similar to the standard service panel that you use to access the ROM operator menus. See "Adding an extra service panel" in Chapter 40, Coin Door.

Connecting to the expansion boards

The main expansion board has a dedicated pin header for the calibration button and its indicator lamp, labeled on the board as "JP3" and "CAL BTN".



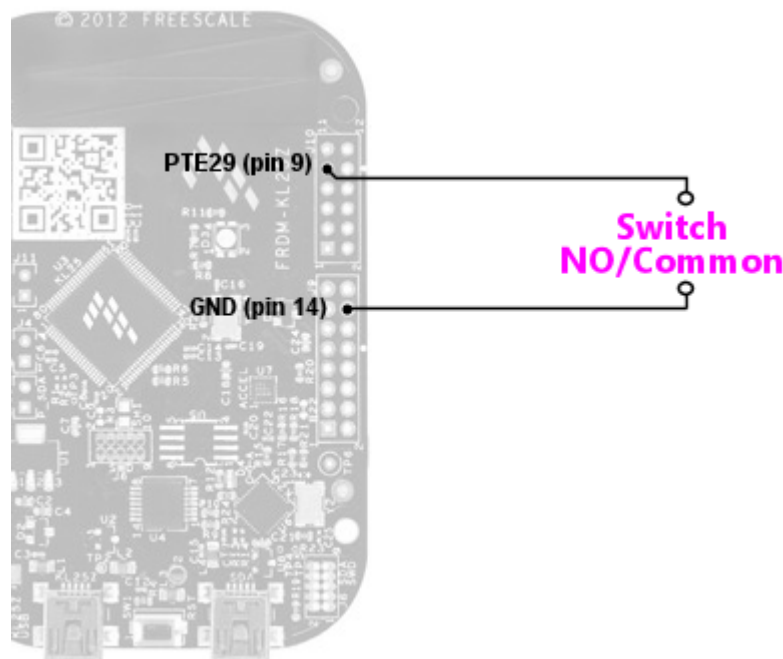


- Connect the Normally Open (NO) and Common terminals on the button to pins 2 and 4 on the header as shown above (it doesn't matter which switch terminal connects to which pin)
- If your button has an indicator LED, connect the "-" terminal on the LED to pin 1 on the header and connect the "+" terminal on the LED to pin 3 on the header

You should connect this through a 2x2 crimp pin housing (included in the parts list). See Chapter 82, Crimp Pins for help with building the housing and connecting the wires. Be sure to mark pin 1 on the housing, to make it easy to remember the orientation whenever you have to plug in the connector. Pin 1 on the housing should line up with the arrow on the board that marks pin 1 on the board.

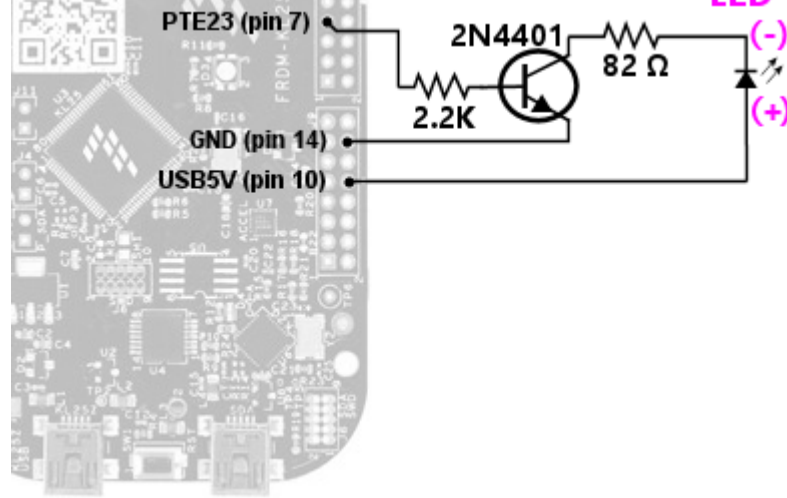
Connecting to the standalone KL25Z

The switch itself is easy. Find the Normally Open (NO) and Common terminals on the switch. Connect these to PTE29 (J10 pin 9 on the KL25Z) and GND (J9 pin 12 or 14). The order of the switch terminals doesn't matter.



The indicator LED, if you want to use one, takes a little more work. You have to build the little transistor amplifier circuit shown below.





Important! The 82Ω resistor was specifically chosen for the blue LED in the illuminated pushbutton in the parts list. If you're using a different type of pushbutton, you'll need to recalculate the resistor value. Look in the button's data sheet to find the "Forward Voltage" and "Forward Current" values for the indicator lamp, then plug those values, along with the 5V supply voltage, into the LED resistor calculator in Chapter 52, LED Resistors. That will tell you the correct value to substitute for the 82Ω resistor. (The $2.2K$ resistor *doesn't* need to be recalculated - it's a constant.)

The GPIO pins (PTE29 for the switch, PTE23 for the indicator lamp) aren't set in stone. PTE29/PTE23 are the defaults that the firmware is initially set to assume, so that's what we're showing in the diagrams. But you can change these to any other GPIO pins you prefer using the Config Tool. To you set different ports:

- Launch the Pinscape Config Tool
- Click on the Settings icon for your device
- Scroll down to the **Plunger calibration button** section
- Select the new ports in the Button Input and Indicator Lamp Output boxes

How to use the calibration button

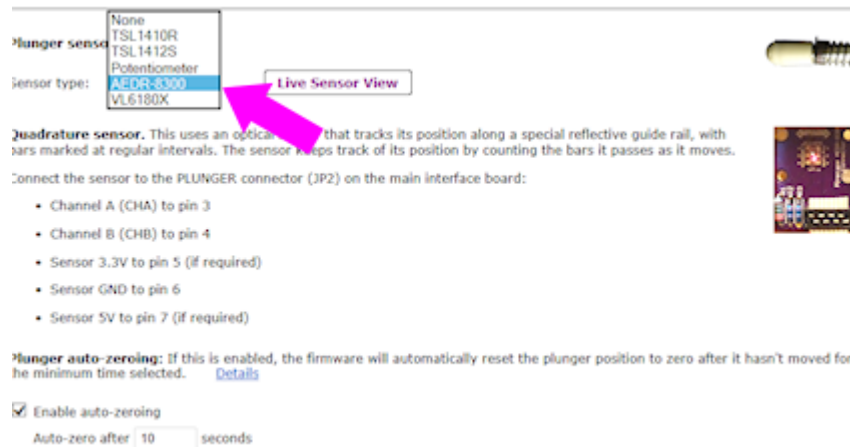
- Make sure the plunger is at its normal rest position
- Press and hold the calibration
- The indicator lamp will flash for about two seconds, then turn solid ON
- If you're not using an indicator lamp, observe the LED on the KL25Z instead: it'll flash blue for about two seconds and then turn solid blue
- When the indicator lamp turns solid ON (and the KL25Z LED turns solid blue), calibration is running!
- You can stop pressing the button now - calibration is active and will run for about 15 seconds
- Pull the plunger all the way back as you normally would during a game, hold it for a couple of seconds, then release it
- Let the plunger come completely to rest
- Repeat the pull-and-release a couple of times
- When the indicator lamp goes out and the KL25Z LED returns to its normal diagnostic flashing pattern, calibration is finished

- The KL25Z LED should flash blue/green to indicate that the plunger has been successfully calibrated

108. Pinscape Config Tool Plunger Setup

The first step in setting up a plunger is to configure the proper sensor type and hardware connections.

- Launch the Pinscape Config Tool
- Go to the Settings page for your device
- Scroll down to the **Plunger Sensor Setup** section
- Select your sensor type from the drop list

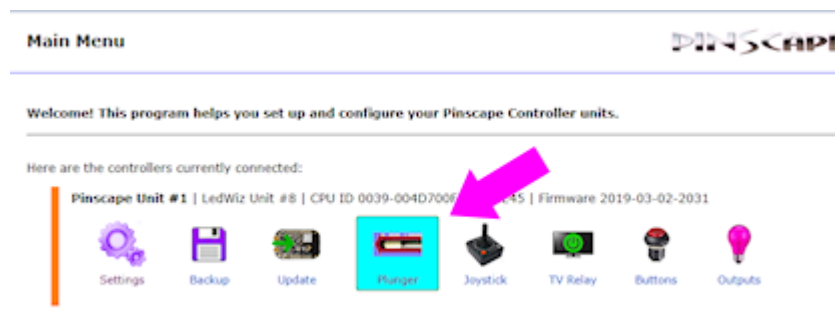


If you're using the expansion boards, the pin connections to the KL25Z should be set automatically. If you're using a standalone KL25Z, you have to set the GPIO pins to match the physical pins where you connected the sensor.

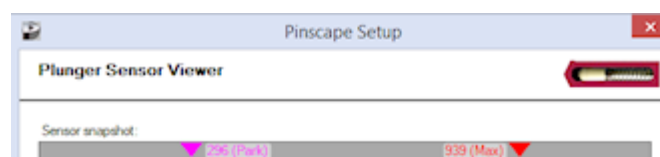
The wiring details are different for each sensor type, and the individual sensor-specific chapters cover that in detail, so refer to the appropriate chapter for the type of sensor you're using for wiring instructions.

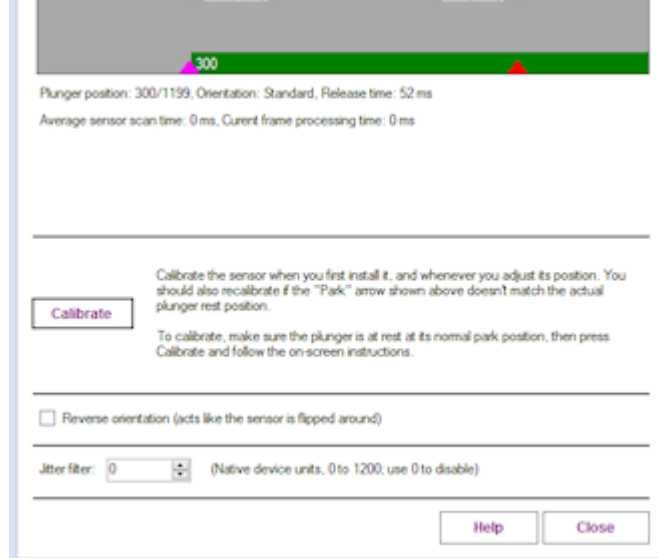
Initial testing

After you've configured the plunger sensor type and pin connections, you're ready to test and calibrate the plunger. Return to the main screen, and click the plunger icon for your device:



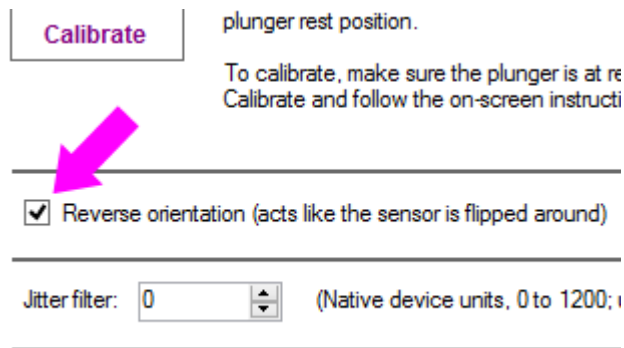
This will bring up the plunger sensor viewer.





The green bar in the "sensor snapshot" area shows the plunger position that Pinscape software detects. The first thing to check is to make sure that this tracks the plunger when you move it. When you pull back the plunger, the green bar should move **towards the right**, and should track the physical plunger motion linearly. The "plunger position" number printed just below should **increase** as you pull the plunger back.

If the bar moves backwards (it moves to the left as you pull back the physical plunger, and the plunger position number decreases), the sensor is installed backwards from the expected orientation. But that's okay! The software can compensate. All you have to do is checkmark the box "Reverse orientation" towards the bottom of the window.



If the bar doesn't move at all, there's something wrong with the sensor or the wiring to the KL25Z. First, go back to the settings page, and double-check that the sensor type and GPIO pin assignments are correct. If that all looks good, it must be a problem with the physical wiring. Check all of the connections: make sure everything is plugged in where it's supposed to be plugged in, and carefully inspect all of the solder joints and crimp pin connections.

Calibrating with the Config Tool

Before you can use the plunger in games, you have to calibrate it, to align the software's numeric readings with the physical range of the sensor. Calibration is done from the sensor viewer window we used above to view and test the sensor operation.

To calibrate:

- Launch the Pinscape config Tool
- Click the Plunger icon for the device to bring up the plunger sensor viewer
- Click the **Calibrate** button
- Follow the on-screen instructions

The calibration process runs for about 15 seconds, during which you should move the plunger over its range per the on-screen instructions. The process stops automatically after the software has collected the sensor range data. The calibration information is saved in the KL25Z's non-volatile memory, so the device remembers it across reboots and power cycles.

The calibration procedure is basically a one-time operation - you only have to do it when you first install the sensor. There's no need to repeat it routinely (for example, you **don't** have to recalibrate every time you reboot Windows or power up the machine). You only need to run through the calibration again if you uninstall and reinstall the sensor. Recalibration is necessary at that point because you'll probably have changed the sensor's physical alignment, at least slightly, so you'll want to recalibrate to get the software back in sync with the new physical setup. You can also recalibrate any time the sensor seems to be off (e.g., the on-screen plunger isn't coming to rest at the normal park position).

Calibrating with the calibration button

If you installed a dedicated calibration button, you can also calibrate using the button instead of going through the Config Tool UI. There's no good reason to do that, as the Config Tool has a friendlier UI that shows more information about what's going on, but the button will work, too, if you prefer to use that for some reason. The procedure is described in Chapter 107, Pinscape Plunger Calibration Button.

Windows calibration = bad!

Windows has its own joystick device calibration procedure, which you get to via the Windows control panel called "Set up USB game controllers". Don't use it!

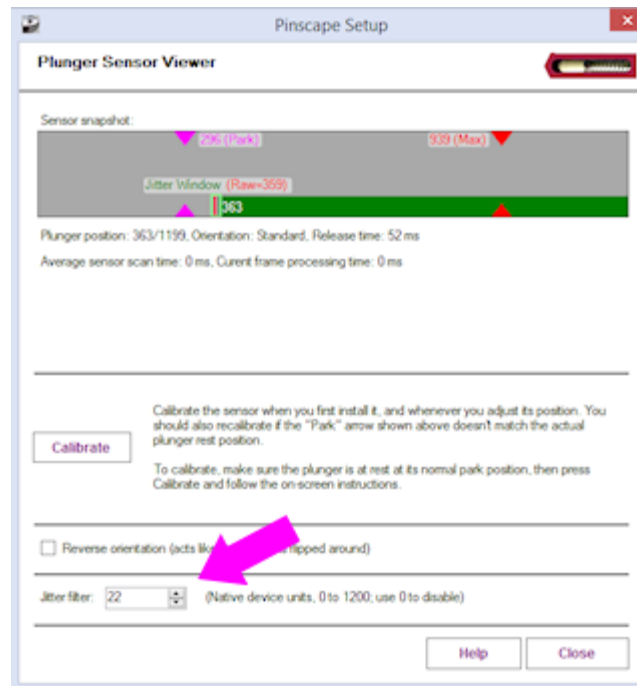
Everyone always wants to run this. They see the calibration option in Windows and think it must be there to help. It *is* there to help, but only for *real joystick* devices. It's a disaster to use with nudge/plunger devices, because they're not anything like real joysticks. Nudge/plunger devices only *pretend* to be joysticks so that they don't need separate device drivers.

If you accidentally ran the Windows calibration before you read this warning (everyone does!), you'll need to delete the Windows calibration. The Windows calibration will screw up the Pinscape readings and make your nudge and plunger inputs act erratically. Fortunately, they made it pretty easy to reset the unwanted calibration data:

- Open the "Set up USB game controllers" control panel (press Windows+R, type **joy.cpl**, press Enter)
- Select the Pinscape device
- Go to the Settings tab
- Click Reset to Defaults

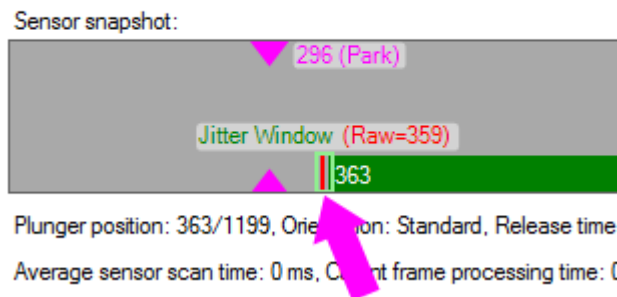
Jitter filter

Near the bottom of the Plunger Sensor Viewer window, there's a setting called "Jitter filter":



This sets the size of the "jitter window." When you set this to a non-zero value, the window will be shown visually in the sensor snapshot area, as a little box around the end of the plunger position bar.

Plunger Sensor Viewer



When the jitter window is set to a non-zero value, raw sensor readings that vary within the window will be ignored. Any time the raw sensor reading goes outside the current window, the window will move far enough to contain the new reading.

The point of the jitter filter is to make the on-screen plunger stand still when the *real* plunger is standing still. That might seem something that should happen anyway, without any filtering, and you're right - it *should* happen anyway! But remember that we're dealing with an electronic measuring device, and all measuring devices have some inherent imprecision. Suppose you're asked to measure the length of a line with a ruler, to the nearest 1/32 of an inch, but the ruler only has markings in 1/8 inches. You plop down the ruler and squint at where the end of the line falls between the nearest 1/8" marks, and you come up with the nearest estimate. Now you hand it over to a friend and ask what they think. Chances are that their reading will be just a little different from yours, since you both had to interpolate between the 1/8" marks. And if you go back and take another reading yourself, you'll probably get a third value. Plunger sensor readings are like that: the sensor takes samples

hundreds of times a second, and each sample has a little of that inherent measuring error, so each successive sample is likely to be slightly different from the last one even when the true position isn't changing. That shows up in the on-screen plunger as jitter. You see those slightly variations from one sample to the next as little motions of a pixel or two on-screen.

If the sensor is accurate enough that those little measurement errors are too small to see in the on-screen plunger, the jitter filter can be set to zero. The AEDR-8300 sensor is that accurate. The "analog" sensors - the potentiometer and the IR distance sensors - tend to have higher inherent measurement errors that make jitter filtering helpful.

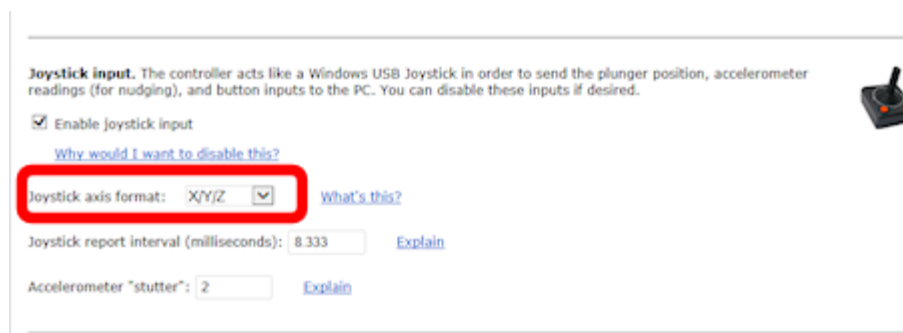
To tune the jitter filter, start by setting it to zero. Observe the green plunger position bar. If it's simply standing still, you're set; you don't need a jitter filter at all. That should be the case with the AEDR-8300 sensor. If it's jittering at all, gradually increase the jitter window size. Observe the light green box around the end of the plunger bar. When the jitter window box is standing still (even though the raw reading, shown by the dark green bar, might still be jittering around within the box), the window is big enough. You can try reducing the window size to see if the green box stays steady with a smaller value.

Smaller is better with the jitter window because a bigger window means less precision when you do actually move the plunger. Filtering like this always trades precision for stability. You want to apply just enough filtering to get rid of the noise (or reduce it to a tolerable level) without filtering out too much of the true information you're trying to measure.

Axis selection

The Pinscape software normally sends plunger readings to the PC using the "Z" axis, since that's the convention that almost all pinball player software uses. I recommend sticking with that default, since it's the most widely compatible option.

However, for special situations, the Config Tool gives you the option to use a different set of axes, known as the "R" axes. This change can be made in the Config Tool's Settings page, under the Joystick section:



For more on the "R" axis settings, see "Axis settings" in Chapter 99, Accelerometer (Nudge) Setup.

Auto-zeroing (quadrature sensors only)

If you're using the AEDR-8300 sensor, you should see a checkbox option to "Enable auto-zeroing", and a box to enter the timing if this is enabled.

This option **doesn't** appear for other sensor types. If you don't see it, it's because

it's not applicable to your sensor.

When auto-zeroing is enabled, the Pinscape software will automatically reset the software plunger position to the "park" position (the point where the plunger comes to rest on its own when you're not moving it) whenever the sensor hasn't detected any motion for the specified time period.

The point of auto-zeroing is to correct for accumulated errors in the software's notion of where the plunger is currently positioned. A quadrature sensor like the AEDR-8300 doesn't actually know *where* the plunger is at any given time; this type of sensor can only detect motion, so it only knows *how far* the plunger has moved from the starting point. These distance measurements are quite precise, but like any sensor, there's always some measurement error, and little errors add up to big errors over time if you just let them keep accumulating. Imagine if you had to measure a mile-long distance with a one-foot ruler: you'd try to carefully line up the starting point and ending point of each one-foot interval, and take a careful measurement each time, but you'd always have a tiny bit of uncertainty in each measurement. After adding up five thousand of those one-foot sections, the tiny errors would add up to at least a few inches, and probably several feet. That's basically how a quadrature sensor measures position: it adds up the net effect of many tiny motions to figure the overall position.

Auto-zeroing helps avoid long-term error accumulation by periodically resetting to a "known state", by assuming that a plunger that's been sitting still for a long period must be sitting at the normal rest position. When there's no motion for a long period (you can specify exactly how long that is via the "Auto-zero after" box), the software simply sets the current internal position counter to equal the park position.

109. ZB Launch Ball

Zeb (aka Steve) of Zeb's Boards came up with a clever feature for his plunger kit that lets the plunger double as a Launch Ball button, for tables like *Medieval Madness* or *Terminator 2: Judgment Day* that were plunger-less in their original arcade versions. The Pinscape software has the same feature, which we call ZB Launch Ball in honor of its inventor. This section explains how to set it up.

The point of the ZB Launch Ball feature is to let you eliminate the need for a separate, physical Launch Ball button on your cabinet, by using your plunger in place of the Launch button for tables that used the Launch button (instead of a plunger) in the original arcade version. Some pin cab builders choose to include *both* a plunger and a Launch Button when designing their cabs, but some people think that looks too cluttered, and would prefer to install just the plunger, like on most of the real machines. If you prefer that cleaner look without the extra button, the ZB Launch feature is really useful, because it lets you operate the tables that require the Launch button even though you don't have a Launch button installed. Plus, even if you install both controls, ZB Launch can still be a nice convenience feature, since many of us are so accustomed to using the plunger that we reach for it reflexively even for games that don't use it. It's nice to have the plunger just work at those times.

How it works

The ZB Launch Ball feature kicks in whenever you're playing table in Visual Pinball that was originally plunger-less. It *only* works with plunger-less tables. It doesn't do anything when you're playing a conventional table with a conventional plunger.

When the ZB Launch feature engages, pulling back and releasing the plunger translates into a momentary press of the Launch Ball button. You can also press the plunger forward slightly to simulate pressing the Launch button. Pressing the plunger forward and holding it there simulates pressing and holding the button, which is useful for a few tables (such as *Championship Pub*) that do something special on long presses of the button.

How does Pinscape know that the table is plunger-less? That's the clever part. Zeb's idea was to use a fake LedWiz feedback device to let the pinball software switch the feature on and off in the plunger controller. When the fake device is "on", the controller knows that ZB Launch Ball is in effect. This fake device is now standard in the online DOF Config Tool: it's the device called "ZB Launch Ball". The DOF Config Tool database has entries for this device all of the plunger-less tables.

Prerequisites

To use ZB Launch, you need DOF to be installed and working, even if you're not using DOF for anything else. DOF is needed because it's what sends the signal to the Pinscape device telling it when ZB Launch Ball mode should be enabled. See Chapter 46, DOF Setup if you haven't already installed it. Follow the instructions to install the DOF software on your PC and set up a Pinscape device using the online DOF Config Tool (configtool.vpuniverse.com).

You'll also have to set up your plunger sensor, configure it, and make sure it's working. Be sure you've gone through the plunger calibration process in the Pinscape Config Tool, because the ZB Launch feature depends upon the plunger resting position being properly calibrated. (**Don't** use the joystick calibration in the Windows joystick control panel. That's a whole different thing that interacts poorly with Pinscape.)

How to set it up

After you have the prerequisites ready, here's how you set up the ZB Launch feature:

- Run the Pinscape Config Tool and go to the Settings page
- Scroll down to the **ZB Launch Ball setup** section
- Make sure the **Enabled** box is checked
- Ignore the "Output port number" field for now
- Select the keyboard key or joystick button you want Pinscape to send to the PC to simulate Launch Ball button presses. The Enter key is the default because that's what VP normally uses. If you've changed VP to use a different key, select that key instead.
- The push distance determines how far forward you have to push the plunger to activate a simulated button press. You can leave this at the default setting for now; you can go back and change it later if this proves to be too sensitive or not sensitive enough.
- Scroll all the way down to the **Feedback device outputs** section, and go to the bottom of the port list. Click the green "+" button in the empty row at the end. This will create a new port of type "Virtual". That means it's not connected to anything physical, which is perfect for the ZB Launch Ball port. Click somewhere in this row. This brings up a box that lets you change the port type. Click "ZB Launch Port". This will add a little red "Launch Ball" button icon in the row. If you scroll back up to the ZB Launch Ball setup section, you'll see that this port has been entered into the "Output port number" field.
- Get out a piece of paper and a **big red marker** and write down the port number you just assigned
- You're all set with the Config Tool setup, so click **Program KL25Z** at the bottom of the window to save the changes
- In your browser, go to the DOF config tool (configtool.vpuniverse.com)
- Go to the Port Assignments page. Select your Pinscape unit from the "Device" drop list.
- Remember the **big red marker** number we wrote down a minute ago? Find that same port number in the list. Open its drop list and select "ZB Launch Ball".
- Click the Generate Config button, download the ZIP file that creates, and unpack it into your DirectOutput folder. This will install updated DOF files with the new port assignment.

Try testing a couple of plunger-less tables as described below. Try a popular table like *Medieval Madness*, since that's likely to have the right default settings already in place in the DOF Config Tool database.

How to test a table

- Load the table in Visual Pinball
- Run the table (press F5)
- Start a game
- When a ball is in the chute, try pulling back and releasing the plunger
- If everything's working, the ball should launch as though you pressed the Launch button

Troll bombs and laser cannons

There are a few tables where the Launch Ball button has special uses in the middle of ball (as opposed to the normal use, of launching the ball from the plunger chute at the start of a ball), such as:

- Firing the cannons in *Star Trek: The Next Generation*
- Using a troll bomb in *Medieval Madness*
- Firing the cannon in *Terminator 2: Judgment Day*

However, by default, the ZB Launch Ball feature doesn't work at most of those odd times. The default DOF Config Tool database is set up so that ZB Launch only works when a ball is in the plunger chute. If you find this annoying (which I do), you'll be pleased to know that you can fix it. It just takes a tiny bit of work customizing your configuration in the DOF Config Tool.

The trick is change the ZB Launch Ball output port setting in the DOF configuration so that it's **always on** while playing the table in question. The default DOF configuration doesn't do that; it only turns ZB Launch on when a ball is in the plunger chute, for a normal start-of-ball launch. That's why the default setup misses cases like the laser cannons in *ST:TNG* and the troll bombs in *MM*. I think it's set up this way out of an abundance of caution, the concern being that the regular plunger action is disabled whenever ZB Launch is enabled. But I personally don't see any downside to leaving ZB Launch enabled full-time for a game like *ST:TNG* that doesn't even have a regular plunger. Leaving it enabled all the time catches all of the special cases like the laser cannons and troll bombs. (You obviously shouldn't do this for games that also need regular plunger input, since it would make the regular plunger unusable, but you wouldn't usually want to use ZB Launch with plunger-equipped tables in the first place.)

Here's the full procedure:

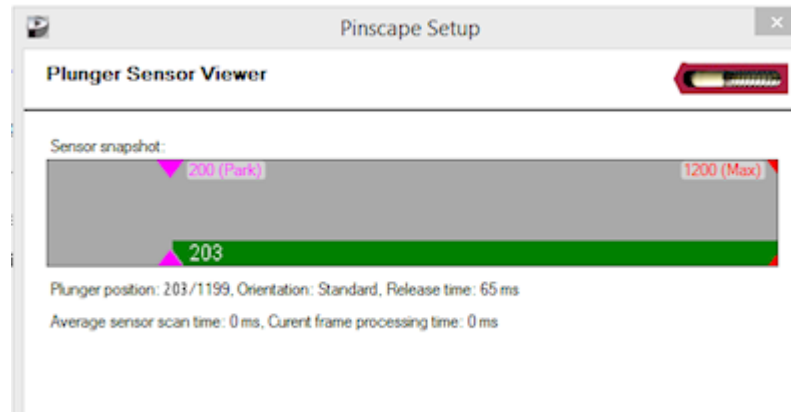
- Open the DOF Config Tool in your browser
- Log in
- Go to the Table Configs tab at the top
- For each table you want to fix:
 - Select the table in the drop list
 - Find the **ZB Launch Ball** row
 - Enter **ON** in the right column
 - Click **Save Changes** at the bottom
- After you've edited all of the tables you want to fix, click **Generate Config** at the bottom to generate the new .ini files; that'll automatically download a ZIP file with the new .ini files when it's done
- Unpack the .ini files from the ZIP file into your DOF Config folder

Troubleshooting

If it's not working, here are some things to try:

- Make sure you **haven't** used the calibration process in the Windows joystick control panel. If you have (or if you're not certain you haven't):
 - Press Windows+R, type **joy.cpl**, press Return

- Find the Pinscape Controller device in the list and double-click it
- Go to the Settings tab
- Click "Reset to defaults".
- Make sure you've gone through the plunger calibration process using the Pinscape Config Tool plunger dialog.
- In the Pinscape Config Tool plunger dialog, check that the on-screen plunger position is displayed at the "Park" position when the actual plunger is at rest. The park position is shown by purple arrows; the current plunger position is the green bar.



- In the Pinscape Config Tool plunger dialog, check that the on-screen sensor readings respond properly when you move the physical plunger.
- Make sure that it's possible to push the plunger forward (against the barrel spring) by about half an inch from the park position. It's important to have a little room for motion forward of the park position, because that's what triggers the simulated button press. In the Pinscape Config Tool plunger dialog, make sure the green bar moves properly (to the left of the purple "park" arrow) when you push the plunger forward.
- The ZB Launch Ball feature depends on DOF, so make sure DOF is working properly with other devices when you run the same table that you're having trouble with. See "Troubleshooting" in Chapter 46, DOF Setup if DOF isn't working.
- The particular table you're running in Visual Pinball must be configured for ZB Launch Ball in the DOF configuration. To check:
 - Open the DOF Config Tool
 - Log in
 - Go to the Table Configs tab
 - Select the table you're playing from the drop list
 - Find the **ZB Launch Ball** box in the right column
 - Make sure there's something in the box
- To get the plunger to activate properly, your DOF device configuration has to be set up to use **ZB Launch Ball**, not the regular **Launch Ball**.
 - Open the DOF Config Tool
 - Log in
 - Go to the Port Assignments tab
 - Select your Pinscape device from the drop list

- Find the **big red marker** port number from the setup procedure
- Make sure it says **ZB Launch Ball**
- Double-check that it **doesn't** just say **Launch Ball** - that's a whole different thing. It has to say **ZB Launch Ball**
- Let's test that the DOF signal is getting through properly:
 - Launch Visual Pinball
 - Load a popular plunger-less table like *Medieval Madness*
 - Don't run it yet!
 - In the VP editor window, click on the **Plunger** button in the left pane
 - Click in an empty area in the middle of the playfield to create a plunger
 - In the Properties window for the new plunger, make sure **Enable Mechanical Plunger** is check-marked
 - Run the table
 - Find the new plunger we created above - it should be sitting out there in the middle of the playfield
 - Try moving your physical plunger back and forth
 - Watch that new on-screen plunger to see if it moves
 - If the new plunger *doesn't* move, that's good. The DOF signal is getting through properly. DOF is working; the problem lies elsewhere.
 - If the new plunger *does* move, the DOF signal is **not** getting sent to the Pinscape device. When the DOF signal is getting sent properly, Pinscape **disables** the regular plunger motion, because it knows that this is a Launch Ball button table that doesn't take regular plunger input. So if the plunger is still moving, Pinscape isn't getting the signal. The thing to focus on is why the DOF signal isn't getting sent properly. Go back through the setup process and double-check all of the port assignments, in both the Pinscape Config Tool and the DOF Config Tool. Go through the DOF troubleshooting steps (Chapter 46, DOF Setup).

110. Pinscape Button Inputs

Once you have a Pinscape Controller set up, it's pretty straightforward to wire your buttons to it.

We're going to assume you've already picked out which buttons you're going to include and that you've installed them in the cabinet. If you're still in the planning stages, and you're figuring out which button functions you want to include, what products to buy, or how to install the buttons in the cabinet, see Chapter 34, Cabinet Buttons.

We discuss button wiring more generally in Chapter 35, Button Wiring, but that section tries to cover all of the different key encoder options, so it's a little less concrete than this chapter. This chapter tries to give you a quick How To guide specifically for Pinscape Controller button wiring, so it should be a little less work to read through.

As with all jobs inside the cab, be sure to do your button wiring work with the power off and the power cord unplugged.

Basic wiring plan

Each button connects to a Pinscape controller with two wires. On the controller, one wire goes to an individual port for the button, and the other goes to the "Ground" or "Common" port. On the button, each wire connects to one switch terminal.

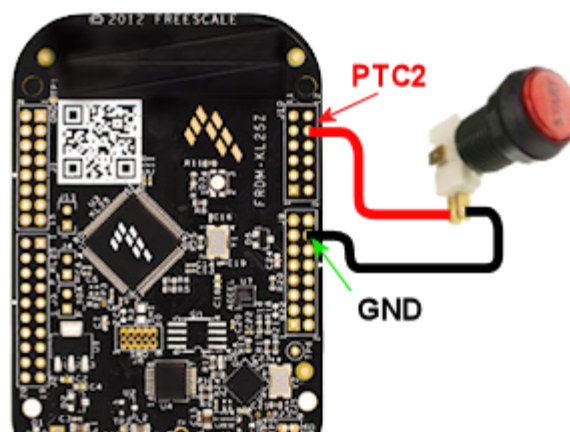
If your button only has two terminals, it doesn't matter which wire connects to which button terminal. If your button has three or more terminals, you do have to identify the correct pair of terminals. Once you do, though, it doesn't matter which wire goes to which of the two. We'll explain how to identify the right pair of connections later in the chapter.

All of the buttons connect to the same Ground/Common port on the controller, so all of the corresponding button terminals end up connected together electrically. This means you can "daisy-chain" the wiring to these terminals, if you wish. More on this shortly.

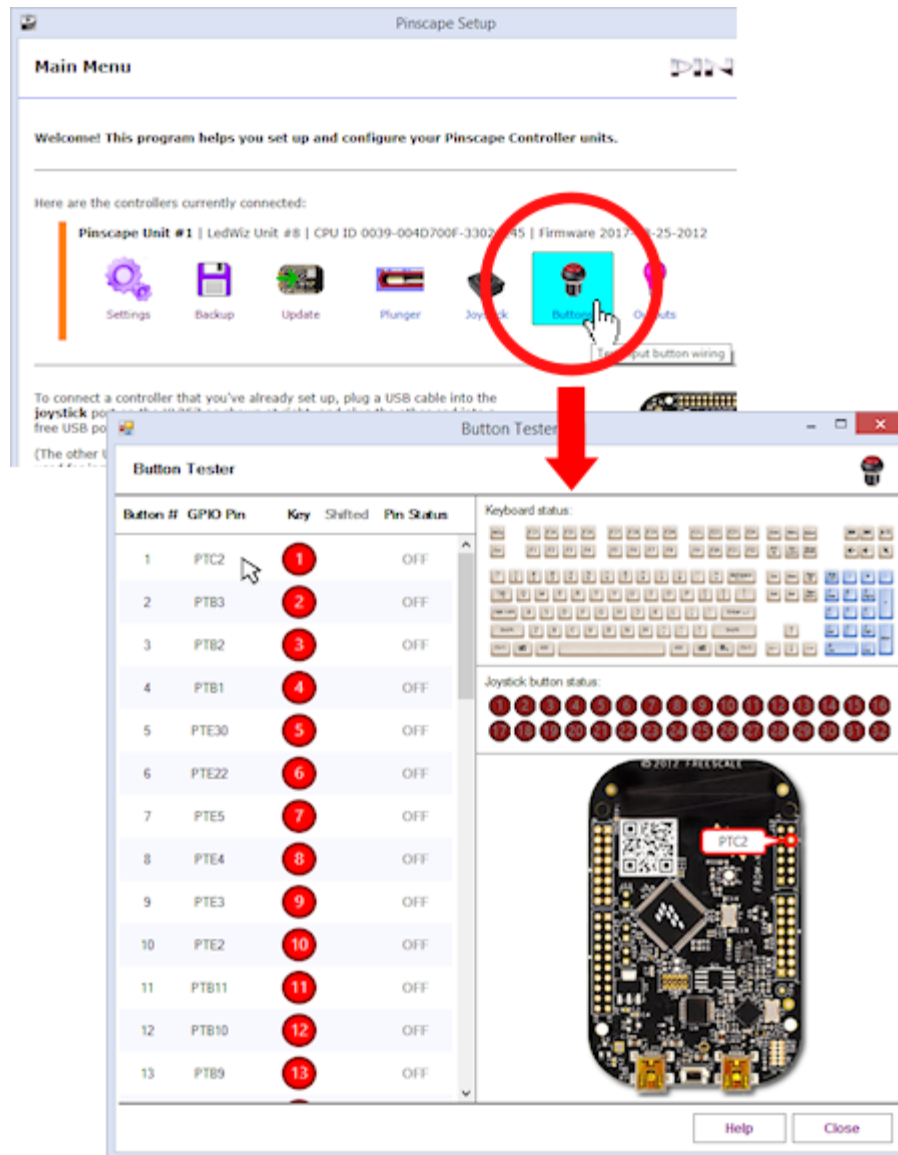
Standalone KL25Z: One wire from each button goes to the "GND" pin on the KL25Z. The other wire goes to the pin for the GPIO port that you assigned to the button in the Config Tool.



Here's an example showing how to wire a button that's assigned to GPIO port "PTC2". For buttons assigned to other GPIO ports, follow the same pattern, simply moving the red wire to the other port's pin. The black wire to the GND pin remains the same.



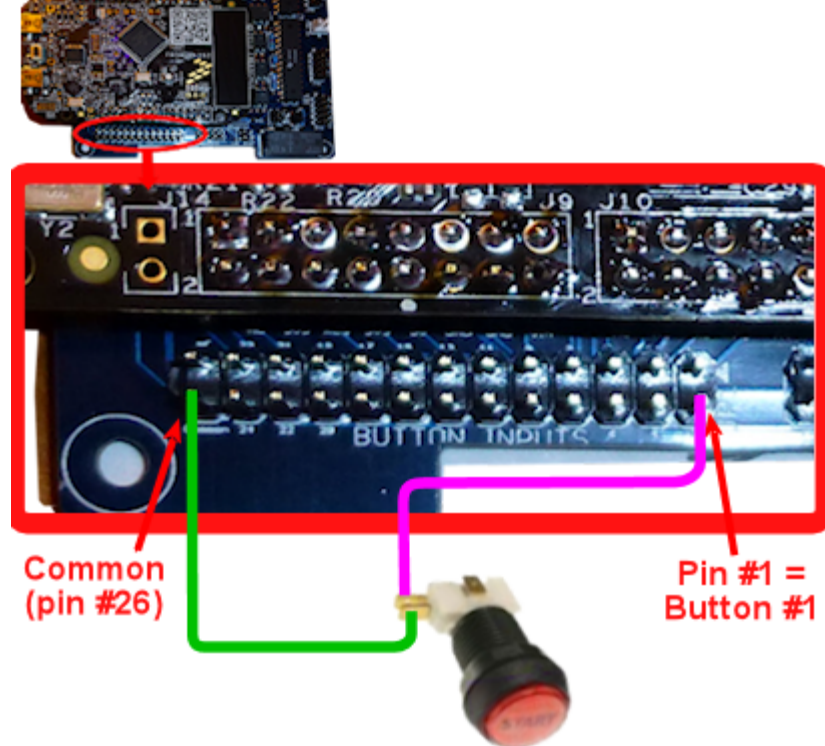
How do you tell which KL25Z pin to use for which button? The Pinscape Config Tool will show you. Run the Config Tool and click on the Buttons icon. This will show you the list of buttons and their assigned ports. Roll the mouse over a button in the list on the left to highlight the pin on the KL25Z diagram. Note that the button assignments are flexible, too: you can use the Settings page in the Config Tool to move buttons to different pins, assign extra pins as buttons, and reassign button pins to other functions.



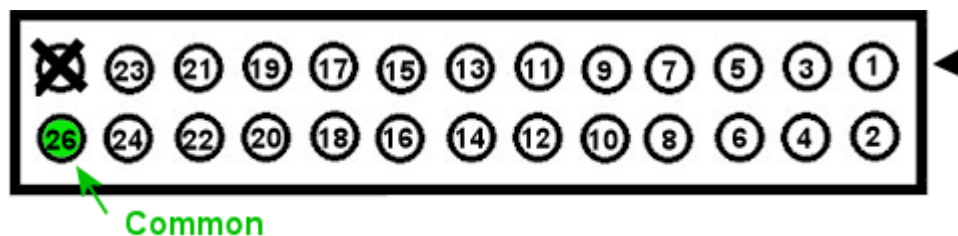
Expansion boards: All buttons connect to the BUTTONS header on the main board. One wire from each button connects to the COMMON pin, #26, and the other connects to the pin that corresponds to the button number.

Here's an example showing how to connect button #1. Follow the same pattern for each additional button; just move the purple wire to the pin corresponding to the new button. The green wire connects to the same Common pin (#26) for every button.





The pin numbers are printed on the board, but they're in tiny type, so here's a cheat sheet. Note that all of the expansion board headers have a little white arrow pointing to pin 1, and pin numbers are always arranged with the even and odd pins in separate rows. (Pin 25 on this header isn't used, so we put a big black X over it here.)



Terminology note: "ground" vs. "common"

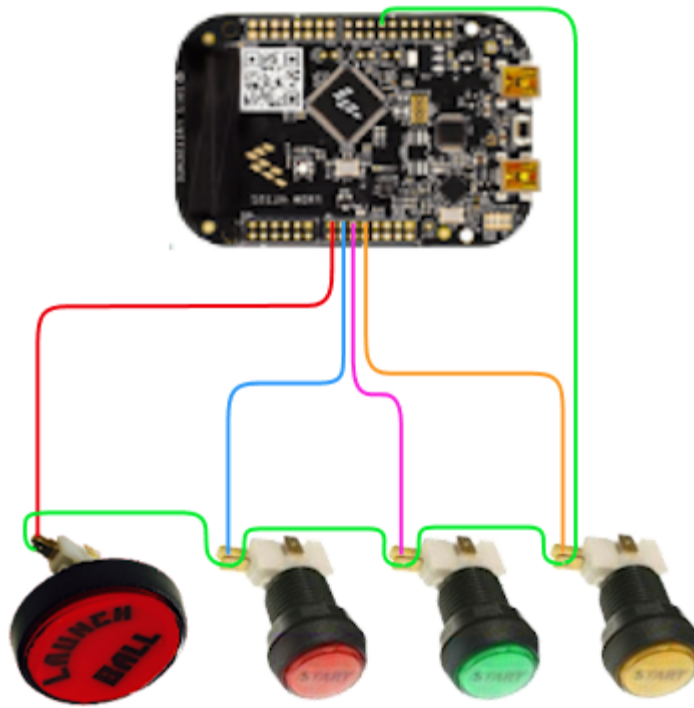
To avoid confusion, it's best to think of the shared wire that connects to all of the buttons as the "Common" terminal, **not** as the "ground" terminal. You might see people refer to this as the ground wire when talking about button controllers in general, but "Ground" is a somewhat confusing term because it has multiple technical meanings that depend on context and aren't always interchangeable (although, to make matters even more confusing, sometimes they are).

Also, be aware that the Common for the Pinscape buttons is **not** the common for non-Pinscape buttons. For example, it's not the common for any button connections on your PC motherboard (e.g., the motherboard Power On or Reset buttons). **Don't** wire any buttons on your PC motherboard or any other non-Pinscape devices to the Pinscape common terminal.

Daisy chaining the common wire

All of the "common" wires on all of the buttons connect to the same terminal on the controller, which means they all end up connected together electrically. This gives you more flexibility in connecting the wire. In particular, it lets you "daisy-chain" the

common wire, by running wires from button to button rather than running a separate wire from each button all the way back to the controller.



Button wiring with a daisy-chained "common" wire (the green wire)

You can also use a mix of daisy-chaining and individual wires to the controller, since in the end, they're all connected together anyway. Do whatever is most more convenient for each button. For buttons that are situated close together, daisy chaining is the way to go, since it uses the least wire. But if a button is off on its own, far away from any others, it might take less wire to connect it directly to the controller "common" terminal than to connect it to another button.

Note that you can only daisy-chain the common, **not** the GPIO port wire. Each button needs its own entirely separate connection to its GPIO port. That's how the controller tells the buttons apart.

How to identify button terminals

Some buttons have a bunch of terminals, so it's not always obvious which ones to connect to the controller. For help with this, see "How to identify button terminals" in Chapter 35, Button Wiring.

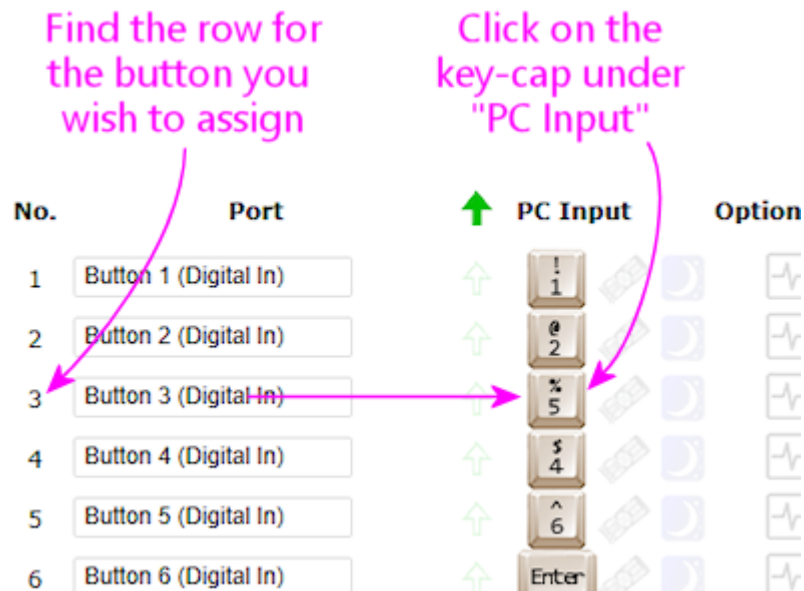
How to assign keys to buttons

What do you do with all of these buttons once you've wired them up? You can assign each one of them to send a keyboard key or joystick button press to Windows. The Pinscape software can emulate a standard PC keyboard, or a joystick, or both, so you can assign any mix of keyboard keys and joystick buttons to your physical pushbuttons.

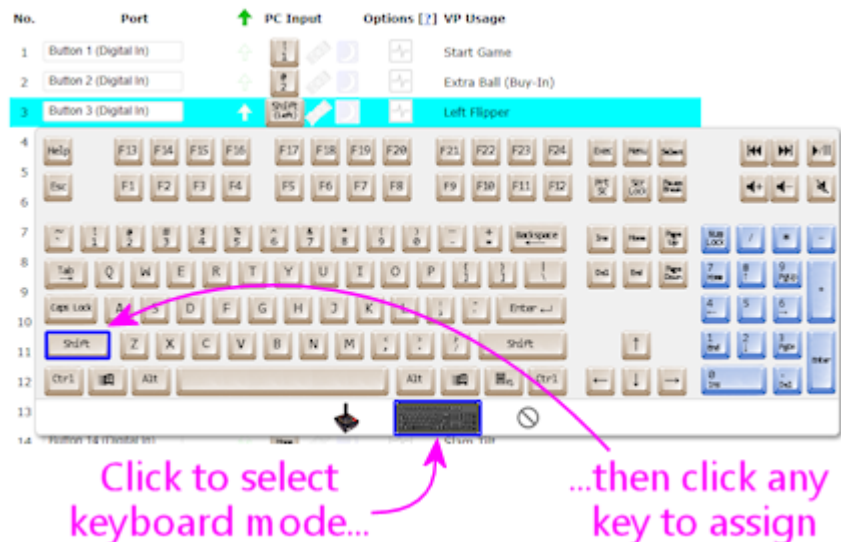
For a list of "standard" key assignments for the main pinball simulator programs, see "Common key assignments" in Chapter 35, Button Wiring.

Here's how you set the keyboard or joystick mappings:

- Open the Pinscape Config Tool
- Go to the Settings for your device
- Scroll down to the **Button Inputs** section
- For each button, click on the little key-cap icon under the **PC Input** column. Each physical button is represented by one row in this list, so just read across the row from the physical button to the key-cap.

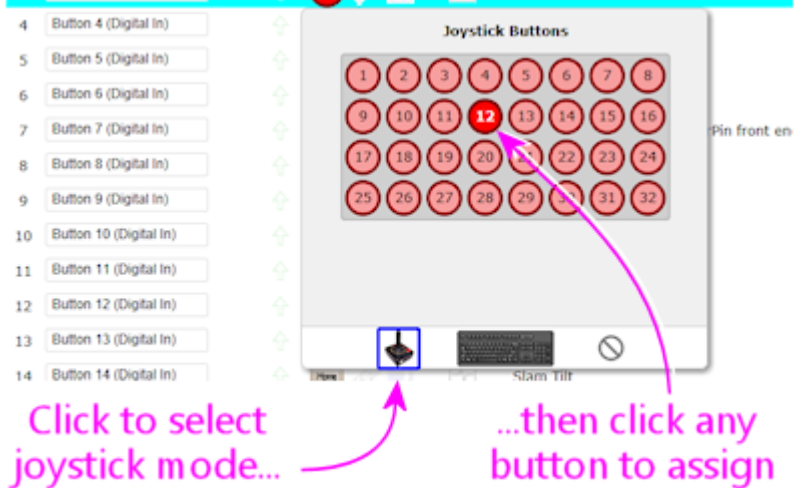


- When you click the key-cap, a pop-up will appear that lets you select a keyboard key or joystick button. To assign a keyboard key, click the keyboard icon at the bottom to switch to keyboard mode, then select the desired key from the mini keyboard diagram.

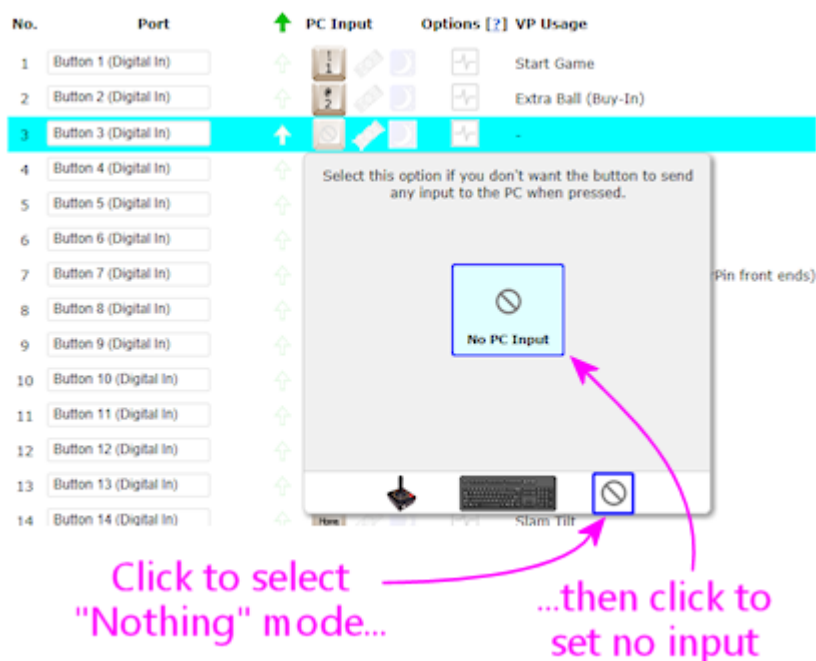


- To assign a joystick button, click the joystick icon at the bottom to switch to joystick mode, then select the desired button. Joystick buttons are simply numbered from 1 to 32. Note that Visual Pinball is limited to 24 buttons, so don't use buttons 25-32 for anything you want to use in VP.



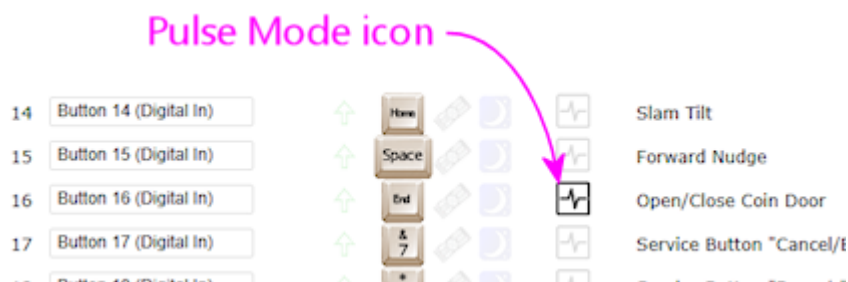


- You can also assign a button to nothing at all, by clicking the "Forbidden" icon at the bottom, then clicking "No PC Input". This just means that the button doesn't send anything to the PC when you press it.



Pulse Mode

Under the Options column for each button, you'll notice a little icon like this:



That icon sets the "Pulse Mode" option for the button.

To understand what Pulse Mode does, it helps if we have a clear picture of what *normally* happens when you press a button. Normally, when a button *isn't* set to Pulse Mode, the button operates as you'd probably expect: pressing the button sends

a "Key Down" signal to Windows telling it that the key is down, and releasing the button sends a "Key Up" signal. In other words, as far as Windows is concerned, the button registers as "pressed" for exactly as long as you're actually pressing it.

Pulse Mode changes that behavior. In Pulse Mode, the button sends an *entire key press and release cycle* to Windows as soon as you press it. That means it sends a Key Down signal, a slight pause, and a Key Up signal. When you release the button, Windows gets *another* complete press/release cycle. During the time in between, when you're holding down the button, Windows thinks the button isn't pressed. As far as Windows is concerned, there's a brief press-and-release when you press the button, and a second brief press-and-release when you release the button. So it's as though you had briefly pushed the button twice, instead of holding it down once.

What's the point of this? It's mostly to accommodate Visual Pinball's *old* coin-door handling, which required a momentary press of the End key when you opened the door, and another momentary press when you closed the door. The normal way to implement the coin door switch physically is with a toggle switch that's ON the whole time the door is open. Pulse Mode was designed to translate that physical switch arrangement to VP's former need for a pulse each time the switch changed from OFF to ON or vice versa. There's actually a better way to handle this now, which is to make some changes in VP scripting so that it can work with the physical coin door ON/OFF switch directly. This is all explained later in this section under "Special handling for the coin door position switch".

"Shift" button

The Pinscape controller lets you give **two meanings** to each button: a "normal" meaning and a "Shifted" meaning. This lets you effectively double the number of commands you can access through your cabinet buttons without adding any more physical buttons. You access the "Shifted" meaning of each button by holding down a designated "Shift Button" while pressing the other button.



Caution! The terminology here can be awfully confusing, because this "Shift Button" feature doesn't have anything to do with the normal SHIFT key on your Windows keyboard. Pay close attention to the words **key** and **button**. **Key** refers to a Windows keyboard key; **button** refers to a physical pushbutton on your cab.

Here's how this works:

- You start by designating one of your regular buttons as the Pinscape Shift Button. You can choose any button you want for this function, and it's perfectly okay to use a button that *already* has a normal function of its own. Let's say we designate the "Extra Ball" button as the Shift Button (that's the one I use on my cab).
- You assign a normal meaning to each button as usual: "Start" sends the "1" key to Windows, "Exit" sends the "Esc" key to Windows
- You even assign a normal meaning to your designated Shift Button, so in our example we assign "Extra Ball" to send the "2" key to Windows
- You can even assign the keyboard SHIFT keys as usual! You know how I said this was going to get confusing? Well, here it is! The Pinscape Shift Button doesn't have anything to do with the Windows keyboard SHIFT keys. So we're still going to assign the Left Flipper button to send the LEFT SHIFT keyboard key to Windows, and we're still going to assign the Right Flipper button to send the RIGHT SHIFT key to Windows.

- You can now *also* assign a *second* meaning, the "Shifted" meaning, to each button *other than* the Shift Button itself
- For example, I use the right flipper and MagnaSave as "shifted" Volume Up and Down keys. To do this, I assign the *second* meaning of my right MagnaSave button to be the "Media Volume Up" keyboard key, and I assign the second meaning of my right Flipper button to be the "Media Volume Down" keyboard key.
- When I want to use my flipper buttons, I just use my flipper buttons. They send the LEFT SHIFT and RIGHT SHIFT keyboard keys as usual.
- When I want to use my Volume Up and Volume Down keys, I press and hold the Extra Ball button (my Shift Button). As long as I'm holding Extra Ball down, all of my other buttons get their second, "Shifted" meanings. So now when I press Right Flipper, I'm sending a Media Volume Down key to Windows instead of a RIGHT SHIFT key.
- How about if I want to send an Extra Ball ("2") key press to Windows? Easy: I just press and release the Extra Ball button. The button only acts like the Shift Button as long as you're holding it down; if you just press it and release it, its normal key mapping is used instead.

How to designate a Shift Button

- Open the Pinscape Config Tool
- Go to the Settings page
- Scroll down to the **Button Inputs** section
- In the "Shift button number" box, enter the number of the button port that you want to use as the Shift Button












Button inputs. You can use the KL25Z as a key encoder to connect pinball-style buttons on your cabine wiring connections and key assignments below. Each input can be mapped as a joystick button or keyboard key assignment to change a setting.

[Test Buttons](#)

↑ **Shift button number:** [Help](#)

☒ Shift OR Key mode
☐ Shift AND Key mode

[Set standard joystick buttons](#) |
 [Set standard keyboard keys](#) |
 [View standard key assignments](#)

No.	Port	↑ PC Input	Shifted	Options [?]	VP Usage
1	Button 1 (Digital In)	↑ 			-/-
2	Button 2 (Digital In)	↑ 			Extra Ball (Buy
3	Button 3 (Digital In)	↑ 			Coin In/-
4	Button 4 (Digital In)	↑ 			Coin In (secon

- Alternatively, just click the ghostly arrow icon in the row next to the button you want to designate












Button inputs. You can use the KL25Z as a key encoder to connect pinball-style buttons on your cabine wiring connections and key assignments below. Each input can be mapped as a joystick button or keyboard key assignment to change a setting.

















[Test Buttons](#)

↑ **Shift button number:** [Help](#)

☒ Shift OR Key mode
☐ Shift AND Key mode
















[Set standard joystick buttons](#) |
 [Set standard keyboard keys](#) |
 [View standard key assignments](#)

No.	Port	↑ PC Input	Shifted	Options [?]	VP Usage
1	Button 1 (Digital In)	↑ 			-/-
2	Button 2 (Digital In)	↑ 			Extra Ball (Buy
3	Button 3 (Digital In)	↑ 			Coin In/-
4	Button 4 (Digital In)	↑ 			Coin In (secon

No.	Port	PC Input	Shifted	Options [?]	VP Usage
1	Button 1 (Digital In)	 			-/-
2	Button 2 (Digital In)	 			Extra Ball (Buy
3	Button 3 (Digital In)	 			Coin In/-
4	Button 4 (Digital In)	 			Coin In (secon







How to tell the difference between the Pinscape "Shift Button" and the Windows "SHIFT key"

This is how the Shift Button looks in the key setup:

Port	PC Input	Shifted
1 (Digital In)	 	
2 (Digital In)	 	
3 (Digital In)	 	
4 (Digital In)	 	
5 (Digital In)	 	

The setting above *doesn't* send a SHIFT key to Windows when you press that button. If you press and release that button, it'll send the "2" key to Windows. Nothing at all to do with the SHIFT key! The green arrow means that this is the Pinscape Shift Button, so if you hold down this button while pressing *another* button, the other button will use its second, "Shifted" meaning.


This is how the Windows SHIFT keys look:

8	Button 8 (Digital In)	 	
9	Button 9 (Digital In)	 	
10	Button 10 (Digital In)	 	
11	Button 11 (Digital In)	 	

When you press one of those keys, they'll send LEFT SHIFT and RIGHT SHIFT keyboard keys (respectively) to Windows.

Shift AND vs. Shift OR modes

Right below the Shift Button Number box on the setup page, you'll notice this cryptic pair of radio buttons:

 **Shift button number:**

☒ Shift OR Key mode
☐ Shift AND Key mode

That lets you control how the Shift Button works when you press it on its own.

"Shift OR Key mode" means that each press of the Shift Button will be act as an invocation of the Shift Button feature, *or* it'll send the Windows key associated with

the button. Never both.

How is this decided? Easy: if you press another key while holding down the Shift button, the Shift button has fulfilled its Shift function and *won't* send its regularly assigned key. If not, it hasn't acted as a Shift Button this time, so it sends its regular key *when you release it*.

This is how I have my Extra Ball button set up. When I use it for its Shift Button function to access my Volume Up and Volume Down buttons, I **don't** want it to send a superfluous "2" keystroke to Windows when I'm done. I just want the Volume Up/Down keys to be sent. But when I just press the button on its own, I do want it to send a "2" key. "Shift OR Key mode" makes it smart that way, so I get the one function I want each time I use the button.

This "smart" action comes at a price, though. It causes a little bit of weirdness in the key press timing, due to that part about sending the normal key *when you release the button*. That can be a little strange, because all of the other buttons send their keys as soon as you press them. But we can't do that if we want the "smart" behavior, since we can't predict whether or not you're planning to press any other keys when you first press the Shift button. We're not mind readers!

"Shift AND Key mode" means that the button always performs both of its functions every time you press it. It starts sending its associated "normal" key assignment as soon as you press it, and it *also* acts like the Shift Button as long as you're holding it down.

Some people prefer this mode because it acts more like a normal button by sending the Windows key press immediately, rather than waiting until you release the button. But you have to give up the "smart" either/or feature to make that possible, so it's a trade-off. If you've assigned this key to something that's ignored most of the time (Extra Ball is actually a pretty good choice for that), you might not mind the superfluous keys that get sent when you're only intending to use the Shift function.

How to assign a second "Shifted" meaning to a key

Once you've designated a Shift Button, you should immediately see a second column for key assignments show up in the Button Inputs list.

Before designating a Shift Button:

No Shift Button assigned yet

One column for key assignments







Button inputs. You can use the KL25Z as a key encoder to connect pinball-style button wiring connections and key assignments below. Each input can be mapped as a joystick or key assignment to change a setting.

[Test Buttons](#)

↑ **Shift button number:** [Help](#)

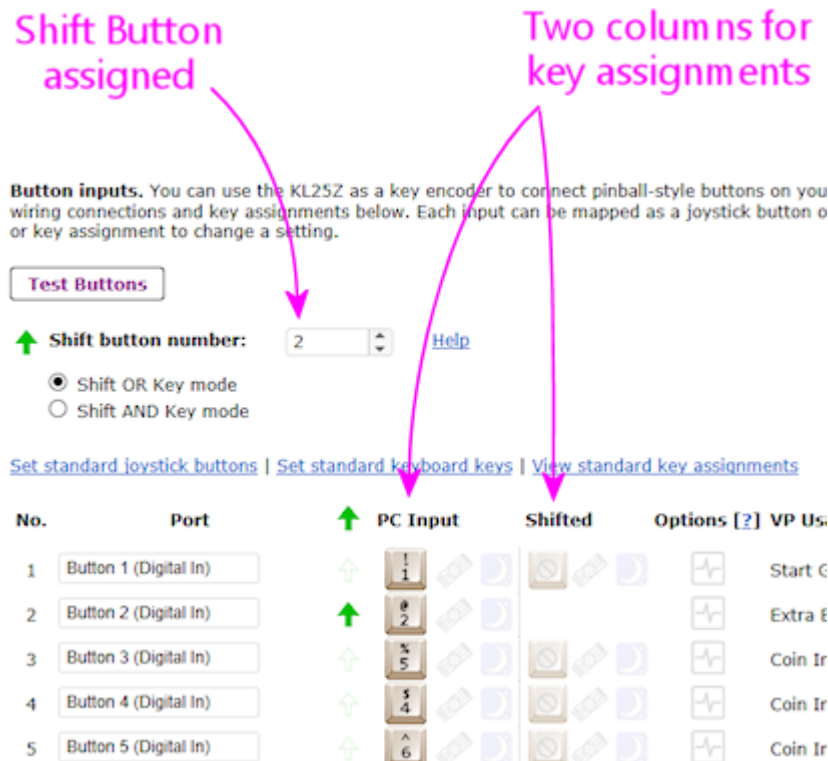
☒ Shift OR Key mode
☐ Shift AND Key mode

[Set standard joystick buttons](#) | [Set standard keyboard keys](#) | [View standard key assign](#)

No.	Port	↑ PC Input	Options [?] VP Usage
1	Button 1 (Digital In)	↑  	 Start Game
2	Button 2 (Digital In)	↑  	 Extra Ball (Bt

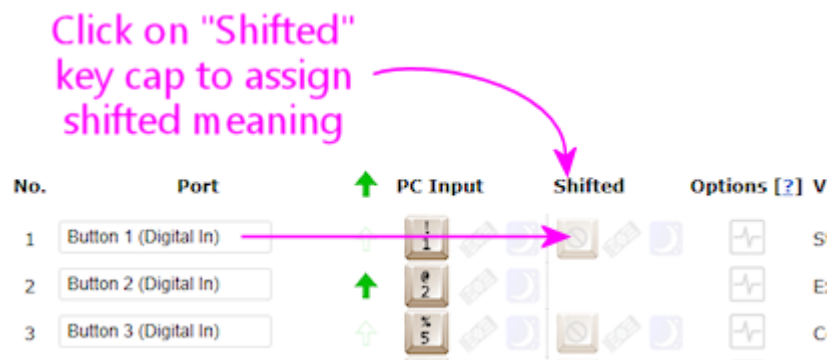


After designating a Shift Button:



The original "PC Input" column is where you enter the normal, un-shifted meaning of the button. The new column that's added when you designated a Shift Button, "Shifted", lets you enter the shifted meaning of the button.

Entering the shifted key assignments is exactly like entering the normal un-shifted key assignments. Just click on the little key-cap image next to the button you want to assign, in this case the *second* key-cap image, the one in the Shifted column. Then select a keyboard key or joystick button from the pop-up.



The Shift Button itself can't be shifted

You might notice that one row is missing that second shifted key-cap icon. Namely, the row for the Shift Button itself. That's because you can't assign a shifted meaning to the Shift Button. There'd be no way you could ever access that shifted Shift Button meaning, since you can't exactly hold down the Shift Button twice at the same time. So the Config Tool just doesn't let you enter a shifted meaning in the first place.

Special handling for the coin door position switch

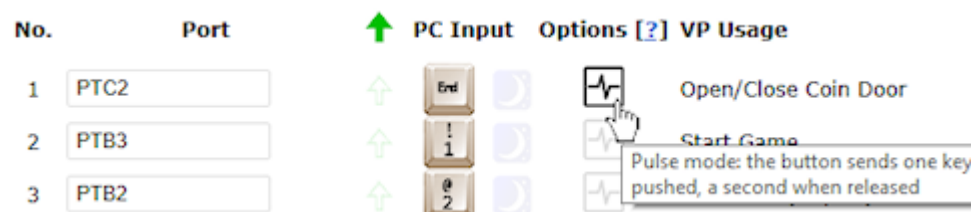
On a real pinball machine, there's a switch that detects when the coin door is open. Pinball ROMs use this to control access to the operator menus, so I'd recommend including one in your build if you're using a coin door. The Chapter 40, Coin Door chapter has suggestions for what kind of switch to use and how to mount it.

Once you have a switch set up, wire its "Common" terminal to the Pinscape button common, and wire its **NC** or **Normally Closed** to a Pinscape button port. Note that this is backwards from most buttons, where you wire the Normally Open terminal. The reason for the reversal is that the geometry of the installation is kind of backwards: when the door is closed, it pushes down on the switch paddle, so the switch is "on". When the door is open, it releases the paddle, so the switch is "off". But we want Closed to read as Off and Open to read as On! The easy way to accomplish this reversal is to use the "other" half of the switch, the Normally Closed side, which reports the opposite status of the Normally Open side.

The coin door open button needs a little bit of special treatment in the software setup. On a real pinball, the coin door switch is just a switch: it's ON when the door is open and OFF when the door is closed. But Visual Pinball, by default, treats it as a toggle button, not a switch: push the button to open the door, push the button again to close the door. There are two options for dealing with this:

- Modify Visual Pinball's core scripts so that VP treats the input as a switch instead of a button. I recommend this approach, as it's simpler and more reliable. See "Setting up the door switch in VP" in Chapter 40, Coin Door for full instructions.
- Use the Pinscape software's "Pulse Mode" option to simulate a toggle button when sending keystrokes to the PC. This doesn't require any change in wiring; everything is done in the Pinscape software. This option is also easy to set up, but I still recommend using the first option (modifying VP's scripts) instead, since it's more reliable. The problem with the toggle key setup is that VP can sometimes miss one of the open/close keystrokes, which makes VP's notion of the coin door's state backwards from reality. It's very difficult to get things back in sync when that happens.

Setting the Coin Door switch to Pulse Mode: If you do want to use the Pulse Mode feature, it's easy to set up. Open the Pinscape Config Tool, go to the Setup screen, and find the slot for the button port you wired to your coin door open switch. Click the Pulse Mode icon:



When this option is selected, Pinscape generates a single key press for the button each time the switch changes from ON to OFF or OFF to ON. That gives VPinMAME exactly what it wants.

Adding a manual Coin Door button: As mentioned above, I recommend avoiding the toggle key setup for the coin door switch, and instead modifying VP's scripts to treat the switch as what it really is, a switch. The big problem with the toggle setup is that VP sometimes misses an open/close key press. This can happen if you open the door right shortly after loading a table, while the table is still initializing, or if you

just do it at the wrong moment while VP is busy. A missed key gets the game into an annoying state where its notion of the coin door is backwards from reality. I ran into this enough times on my own machine that I got tired of fighting it and added a manual pushbutton that also sends the Coin Door key to the PC. I positioned this just inside the coin door so that I can use it as needed whenever VP gets out of sync.

If for some reason you want to set up your coin door switch in toggle mode despite the drawbacks, I'd recommend adding your own manual button like I did. Just set up one more physical button, wired to a separate Pinscape button port. Assign that button port to send the End key *without* Pulse Mode.

111. Pinscape Feedback Outputs

Extensive coverage of the various feedback toys, including what to buy and how to wire them, can be found in Feedback Devices. That includes sections on wiring the Pinscape feedback output ports:

- Chapter 48, Pinscape Outputs Setup (Expansion Boards)
- Chapter 49, Pinscape Outputs Setup (Standalone KL25Z)

112. IR Remote Control

The Pinscape software is equipped to send and receive IR remote control codes. This is primarily for the sake of the Chapter 114, TV ON feature, which can power up your TV(s) at system startup by sending them the necessary IR codes. But you can also use it to transmit any remote control codes you wish at any time, and to translate received IR codes into Windows key presses, allowing a remote control to effectively serve as extra "buttons" on your cabinet.

There are two completely separate hardware projects here: an IR transmitter, which sends out the codes to your TV, and an IR receiver, which can receive codes from your remote control in order to learn the codes specific to your TV or to accomplish those button-press capabilities we mentioned.

The expansion boards include both the IR transmitter and receiver as standard features. You can also build them as add-ons for a standalone KL25Z with a small amount of external circuitry.

Wiring the transmitter and receiver

An IR transmitter isn't anything too exotic. It's really just an LED that emits infrared light. Consumer electronics use a pretty standard range of IR wavelengths, so all we have to do is select an IR LED that emits light of the right "color".

Physically, an IR LED looks like other small LEDs. The only obvious difference is that the case is often opaque or dark plastic rather than the clear plastic you'd find in an optical-wavelength LED.



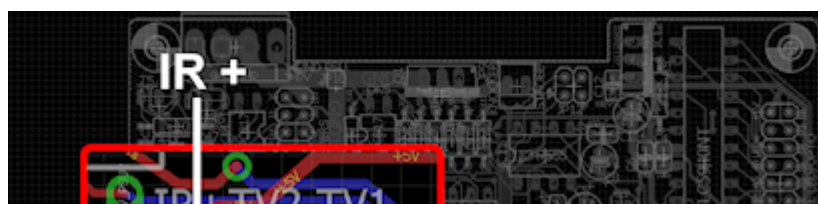
The Pinscape software controls an IR LED through a GPIO port on the KL25Z. The software rapidly turns the LED on and off to form a pattern of flashes that your TV interprets as a remote control command.

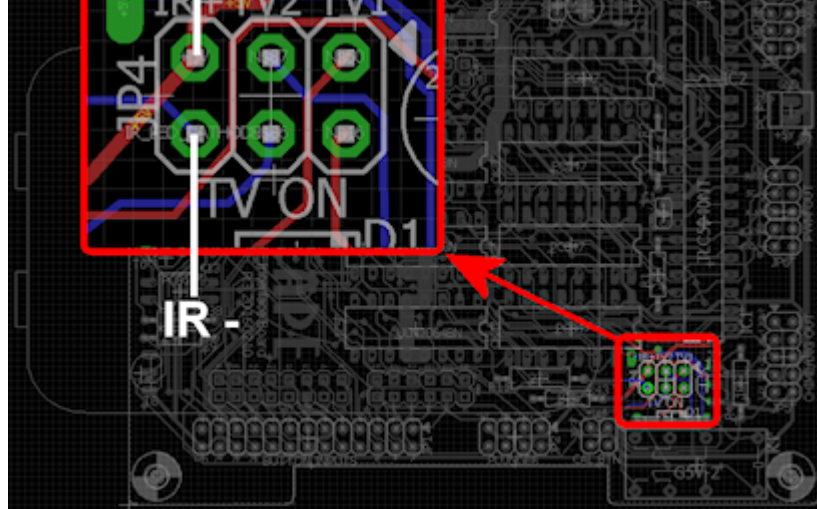
An IR receiver is a more specialized device, purpose-built to read and demodulate signals from an IR remote transmitter. But fortunately they're quite cheap and easy to connect. They're actually easier to connect electrically than an IR LED is, because they don't need much in the way of additional parts.

Connecting an IR emitter to the expansion boards

The IR transmitter LED connects to the "TV ON" pin header on the main expansion board, also labeled JP4. This header has six pins; the IR LED connects to the two pins in the column labeled **IR+**.

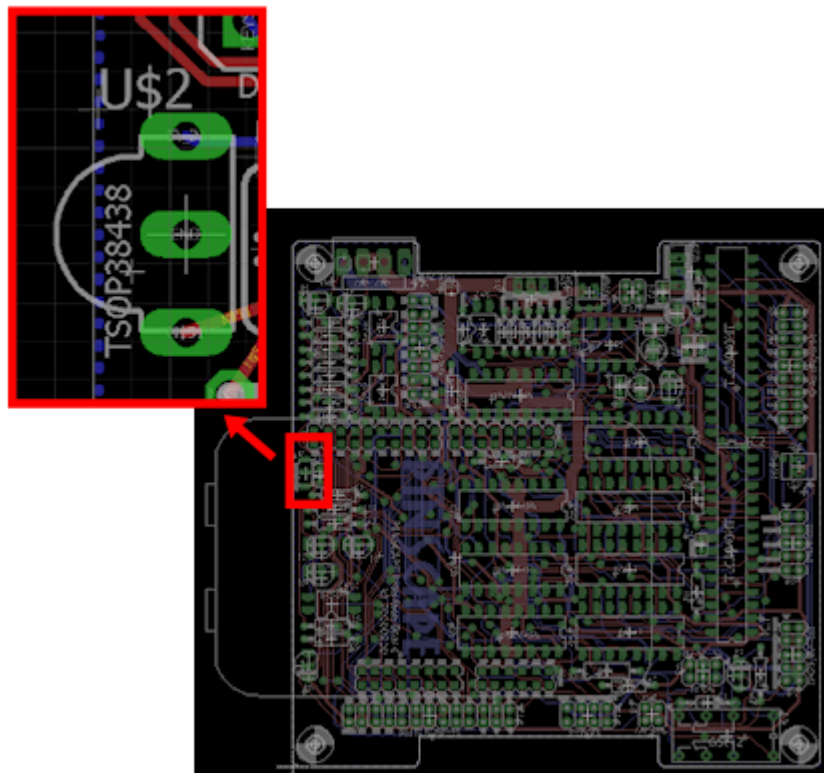
You don't have to connect anything else to these pins besides the IR LED. Just connect the LED so that its **long leg** connects to the **IR+** pin, and the **short leg** connects to the **IR-** pin.





Connecting an IR receiver to the expansion boards

Simply install the IR receiver chip directly in its spot in the expansion boards. You'll need to bend the legs at a 90° angle to that it doesn't stick out too far, as it nestles in under the KL25Z.



No expansion boards: using a pre-built kit

If you're not using the expansion boards, and you'd rather not build any external circuits by hand, there are some ready-made kits available. The ones I've seen are designed for Arduinos, which is another microcontroller (similar to the KL25Z, but different hardware) that's popular with robotics hobbyists. I haven't tested any kits myself, but I've heard from a couple of people who have successfully used this kit:

- Amazon.com: MagicW Digital 38khz Ir Receiver 38khz Ir Transmitter Sensor Module Kit for Arduino Compatible

If that product link isn't working, you might try searching for similar products on Amazon or at hobby robotics vendors like Pololu or Adafruit. The transmitters and receivers for IR remotes are fairly standardized, so there's a fairly good chance that

similar kits will be compatible with the KL25Z electronics and the Pinscape software.

Wiring the receiver:

- Connect **VCC** on the receiver board to one of the 3.3V pins on the KL25Z (pins 4 or 8 on J9)
- Connect **GND** on the receiver board to one of the GND pins on the KL25Z (pins 12 or 14 on J9)
- Connect **DATA** on the receiver board to the GPIO pin you've selected for input. Any interrupt-capable GPIO pin can be used, which means that you can use any PTAx or PTDxx port. Configure the port you select as the IR receiver pin in the Pinscape Config Tool settings. See Selecting GPIO pins below.

See Appendix 1, KL25Z Pin Out for the KL25Z pin wiring diagram.

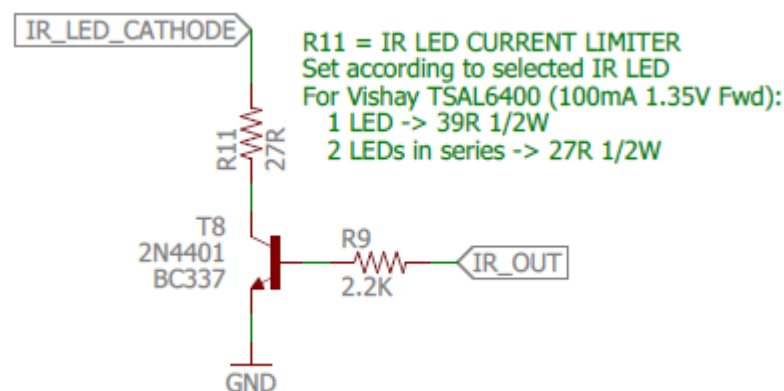
Wiring the transmitter:

- Connect **VCC** on the transmitter board to one of the 3.3V pins on the KL25Z (pins 4 or 8 on J9)
- Connect **GND** on the transmitter board to one of the GND pins on the KL25Z (pins 12 or 14 on J9)
- Connect **DATA** on the receiver board to the GPIO pin you've selected for the transmitter output. A PWM-capable port must be used. Configure the port you select as the IR transmitter pin in the Pinscape Config Tool settings. See Selecting GPIO pins below.

See Appendix 1, KL25Z Pin Out for the KL25Z pin wiring diagram.

No expansion boards: Build your own IR transmitter

The recommended IR LED (TSAL6400) is relatively powerful as LEDs go, so the KL25Z can't drive it directly from a GPIO port. A small power booster circuit is needed. All you need is a simple transistor amplifier. Here's the schematic:



This is the same circuit used on the expansion boards (the diagram above is actually taken from the expansion board schematics).

- Connect the **short leg** of the LED to the wire labeled **IR_LED_CATHODE**. The convention for all LEDs is that the long leg is (+) and the short leg is (-).
- Connect the **long leg** of the LED directly to the +5V power supply.
- Connect the wire labeled **IR_OUT** to the GPIO port on the KL25Z that you're using for the IR output. See Selecting GPIO pins below.
- Connect GND to one of the ground pins on the KL25Z. See Appendix 1, KL25Z

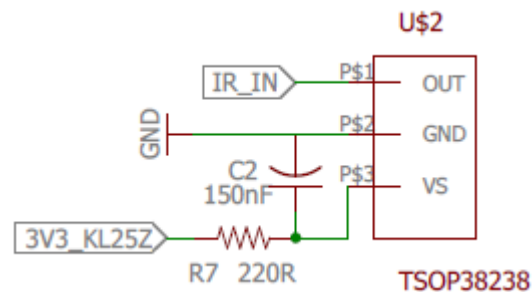
Selecting resistor R11

Choose the resistance value for R11 according to whether you're using one LED or two:

- **For one LED**, use a 39 Ω , ½ Watt resistor
- **For two LEDs**, use a 27 Ω , ½ Watt resistor

No expansion boards: Build your own IR receiver

The recommended IR receiver device, TSOP38238, contains both the optical sensor and the demodulation circuitry. You just need two additional parts (a resistor and a capacitor) to connect it to the KL25Z. Here's the connection diagram:



- Connect **IR_IN** to a GPIO port on the KL25Z that you're using for the IR input. See Selecting GPIO pins below.
- Connect **3V3_KL25Z** to one of the 3.3V pins on the KL25Z. See Appendix 1, KL25Z Pin Out.
- Connect GND to one of the ground pins on the KL25Z. See Appendix 1, KL25Z Pin Out.

Selecting GPIO pins for the power sensing circuit

The IR transmitter can use any PWM-capable GPIO pin on the KL25Z. The default for the expansion boards is PTC9, but you can use any pin with PWM capability. For a list of PWM pins, see Appendix 1, KL25Z Pin Out. The Config Tool will also show you the available pins if you go to the IR section on the Settings page and click on the "IR LED (transmitter) pin" box:

IR Remote Control. The controller can send and/or receive IR remote control signals if you attach some additional components. This can be used with the [TV_ON](#) feature to turn your cabinet TVs on via IR commands at system startup. See the Build Guide for details on the components required and how to connect them. If you don't have any IR components attached or wish to disable them, simply set the pin assignments here to "Not Connected".

IR LED (transmitter) pin: PTC9

Requires a PWM-capable pin

IR receiver input pin: PTA13

Requires an interrupt-capable pin



The IR receiver can use any interrupt-capable GPIO pin. The default for the expansion boards is PTA13, but you can use any pin in the "PTAxx" or "PTDxx" groups. See Appendix 1, KL25Z Pin Out for all interrupt-capable pins, or click on the "IR receiver input pin" box in the IR section on the Settings page in the Config tool:

IR Remote Control. The controller can send and/or receive IR remote control signals if you attach some additional components. This can be used with the [TV_ON](#) feature to turn your cabinet TVs on via IR commands at system startup. See the Build Guide for details on the components required and how to connect them. If you don't have any IR components attached or wish to disable them, simply set the pin assignments here to "Not Connected".

IR LED (transmitter) pin: PTC9

Requires a PWM-capable pin

IR receiver input pin: PTA13

Requires an interrupt-capable pin



Positioning the IR transmitter LED

Assuming you're using the IR transmitter to control a TV, it's best to place the LED as close as possible to the IR receiver window on the TV. You can run as much wire as necessary between the LED and the expansion board port (or your own circuit) to position the LED properly.

Using two IR transmitter LEDs

You can connect two IR LEDs instead of just one. This can be useful if you need to control two TVs, since it lets you position a separate transmitter near each TV's IR receiver. To connect two LEDs, connect them in series:

- Connect the +5V supply to the first LED's long (+) leg.
- Connect the first LED's short (-) leg to the second LED's (+) leg. You can run as much wire between the two as needed to position the two LEDs properly.
- Connect the second LED's short (-) leg to the **IR_LED** connection in the schematic.

Learning IR commands from your remotes

Once you have the hardware installed, you can use the Pinscape Config Tool to learn IR command codes from your remotes. Once Pinscape learns a command code, you can use it for various functions:

- Transmit it at system startup to power up your TV
- Transmit it when you press a button
- Send a key press when the IR receiver receives the code

To learn IR codes:

- Launch the Pinscape Config Tool
- Go to the Settings page
- Scroll down to the **IR Remote Control** section
- Under that, you'll find the **IR Command List** section:

IR Remote Control. The controller can send and/or receive IR remote control signals if you attach some additional components. This can be used with the [TV ON](#) feature to turn your cabinet TVs on via IR commands at system startup. See the Build Guide for details on the components required and how to connect them. If you don't have any IR components attached or wish to disable them, simply set the pin assignments here to "Not Connected".





IR LED (transmitter) pin: *Requires a PWM-capable pin*
IR receiver input pin: *Requires an interrupt-capable pin*

IR Command List. You can program the Pinscape unit with remote control commands that it can then send and receive. Each slot below stores one code. [Help](#)


#	IR Code	TV ON	Key	Description
1	<input type="text" value="1.2.38C7629D"/>			<input type="text"/>
2	<input type="text" value="1.2.38C79D62"/>			<input type="text"/>
3	<input type="text" value="1.2.FFC79D62"/>			<input type="text"/>
4	<input type="text"/>			<input type="text"/>

the limit.

- The IR Command List shows a list of the commands that Pinscape has learned. Initially, this will be empty, so it will just show one empty row.
- To learn the first code, click the "learn" icon () in the empty first row. This will bring up another window that will lead you through the programming process.
 - Get your IR ready, pointing at the Pinscape IR receiver
 - Click the **Learn** button in the dialog
 - Press and hold the button on the remote that you want to learn
 - The "learn" window will show a graphical representation of the learned code, showing the series of on/off light pulses used for the raw bits of the code. If it captures a code successfully, click Save to store the code.
- After you've entered a code into the first blank row, a new blank row will appear. You can continue adding more codes the same way. The "learn" button () programs the code for the row that you click on, so you can go back and replace any row's code with a new code later if you want.
- You can enter a description for each code (such as the name of the remote control button it's associated with) in the Description field. This is just to help you keep track of which buttons you've programmed, so you can enter anything there that will help you identify the button in the future.
- Remember to click "Program KL25Z" at the bottom of the Settings page when you're done, to save the settings in the device.

Using a code to power on a TV

The IR remote feature was primarily designed to be used with the Chapter 114, TV ON feature that powers up your TVs at system startup, so the remote control code list has a special provision to use a code with the TV ON system.

After you've programmed a code, you'll notice that there's a little TV ON icon () in the row next to the code. Just click this icon to activate the TV ON feature for the code. When this icon is activated, Pinscape will automatically send the code at system startup, after a delay that you can program in the TV ON section in the Settings page.

How to transmit an IR code via a Windows command

In your Pinscape Config Tool install folder, you should find another utility program called **PinscapeCmd.exe**. You can use this program to make Pinscape transmit any of the IR command codes it knows at any time via the Windows command line.

- Open a CMD prompt
- Go to the Pinscape Config Tool folder: **CD /D C:\PINSCAPE** (or wherever it's installed)
- Type **PinscapeCmd SendIR=1**

The **SendIR** command transmits the learned command code in the given slot. If you go back to your IR command list, the number corresponds to the row number in that list.

You can put that command in a .BAT script as well, so you can send IR commands from any programs that can launch .BAT scripts, such as PinballX or Pinbally, or automation programs like AutoHotKeys.

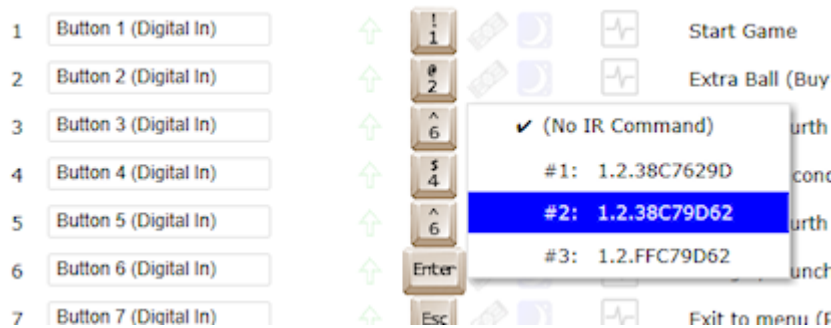
How to transmit an IR code when you press a button

You can program Pinscape to send an IR code when you press a cabinet button. This might be useful if you want to set up a hidden button that turns your TV on or off, for example.

- Open the Config Tool
- Go to the Settings page
- Scroll down to the Button Inputs section
- Click the Remote Control icon in the row for the button you want to assign to an IR command



- Select the learned IR command code to send from the list



- Click **Program KL25Z** to save settings
- Now whenever you press that button, the associated IR code will be transmitted

How to send a Windows keyboard key press when an IR code is received

You can also program Pinscape to send a key press to Windows when the IR receiver sees a given IR code. This lets you turn a remote control into a set of additional "buttons" that you can use to access Windows functions, without festooning your cabinet with even more physical buttons.

To make this work, you'd have to situate your IR receiver where it can "see" IR codes from your remote. The expansion boards don't really make a provision for this, since they situate the receiver right on the main board, where it'll probably be hidden away inside the cabinet. I guess you could open the coin door and point the remote inside. If you really want to use this feature regularly, though, you'd have to come up with a fancier setup where you move the IR receiver somewhere more accessible, like behind an opening in the front of your cabinet.

Aside from those complications with physically getting the IR signal to the receiver,

it's easy to set up the IR-to-key-press feature:

- Open the Config Tool
- Go to the Settings page
- Scroll down to the IR Remove Control section
- Click on the key-cap icon for the command you want to associate with a key press
- That will show the same key/joystick selection popup that's used to map buttons to key presses, so just select a keyboard key or joystick button in the usual fashion
- Click **Program KL25Z** to save settings
- Now whenever the IR receiver receives that code, Pinscape will send the associated key press to Windows

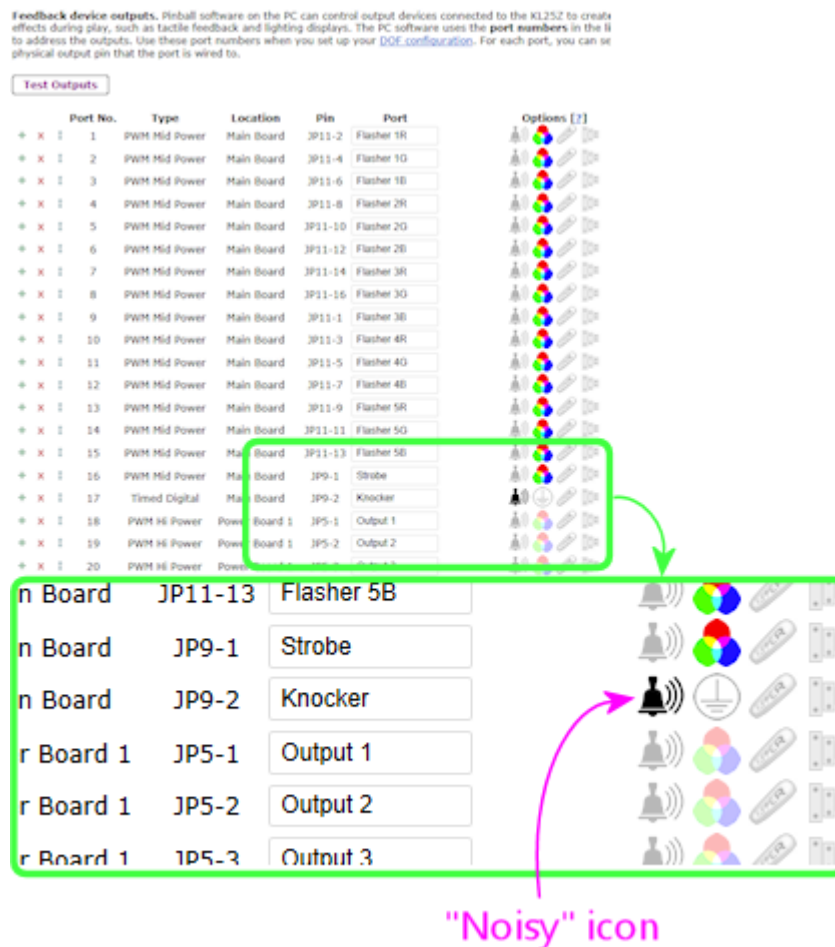
113. Pinscape Night Mode

Night Mode lets you tell the Pinscape controller to shut off the noisy feedback devices in your cabinet, so that you can play games more quietly during late night hours without disturbing neighbors or housemates who are trying to sleep. You can designate specifically which feedback devices are noisy, so that the purely visual devices like flasher lights remain enabled.

Designating the noisy devices

Night Mode only disables the feedback output ports that you specifically mark as "noisy" in your Pinscape settings. To select which ports are noisy:

- Open the Pinscape Config Tool
- Click on the Settings icon for your controller
- Scroll down to the **Feedback device outputs** section
- Click on the "bell" icon for each port that you want to mark as noisy



Activating Night Mode

There are several ways to switch in and out of Night Mode:

- You can control the mode using commands on Windows.
- You can set up a dedicated hardware pushbutton or toggle switch just for Night Mode. This is a perfect kind of button to put on a custom inside-the-coin-door service panel (see "Adding an extra service panel" in Chapter 40, Coin Door). The bottom of the cabinet also works well for this.

- You can designate a "shifted" button as the Night Mode toggle - one that you're already using as a regular button to send a key to the PC, but "shifted", meaning you press it in combination with another button to give it a second function.

Controlling Night Mode via Windows commands

Go to the directory where you installed the Pinscape Config Tool, and you'll find another program there called **PinscapeCmd.exe**. This is a utility program that you can invoke from the CMD prompt to send commands to the controller. You can also use it in batch (.BAT) scripts.

To use the PinscapeCmd program to switch in and out of Night Mode:

- Open a CMD prompt window (Windows+R, type **CMD.EXE**, press Enter)
- CD to the Pinscape folder (e.g., **CD /D C:\PINSCAPE**)
- To turn Night Mode ON, type **PinscapeCmd NightMode=ON** and press Enter
- To turn Night Mode OFF, type **PinscapeCmd NightMode=OFF** and press Enter

The capitalization isn't important. We just showed mixed case for clarity.

Note: if you have more than one Pinscape controller device installed in your system at the same time, you have to specify which one you want to send the command to, by adding **Unit=number** to the command before the NightMode option:

```
PinscapeCmd Unit=2 NightMode=ON
```

You can enter the same syntax in a Windows .BAT script. That lets you control Night Mode from any program that can launch a .BAT script, such as a front-end program like PinballX or Pinbally, or an automation program like AutoHotKeys.

Setting up a dedicated Night Mode button

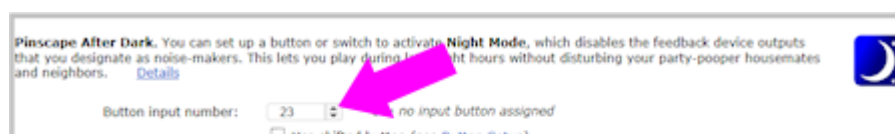
If you want to use a dedicated button or switch to control Night Mode, simply select a pushbutton or toggle switch, and wire it to a Pinscape button input port just like any other button. See Chapter 110, Pinscape Button Inputs for general wiring instructions.

You can use either a pushbutton or an on/off switch:

- If you use a pushbutton (meaning a button with a switch that only turns on for as long as you're pressing the button, like the Start button or a flipper button), it will *toggle* Night Mode on or off each time you push it.
- If you use a toggle switch, such as a sliding switch or rocker switch that stays on until you turn it back off, the Night Mode status will simply follow the switch position. Turn the switch ON, and Night Mode will be on; turn the switch OFF, and Night Mode will be off.

The wiring is the same for either kind of button.

Once you've wired the button to a port, you just have to tell Pinscape about it. Run the Pinscape Config Tool, go to the Settings page, and scroll down to the **Pinscape After Dark** section. Enter the port number where you wired the Night Mode button in the **Button input number** box.

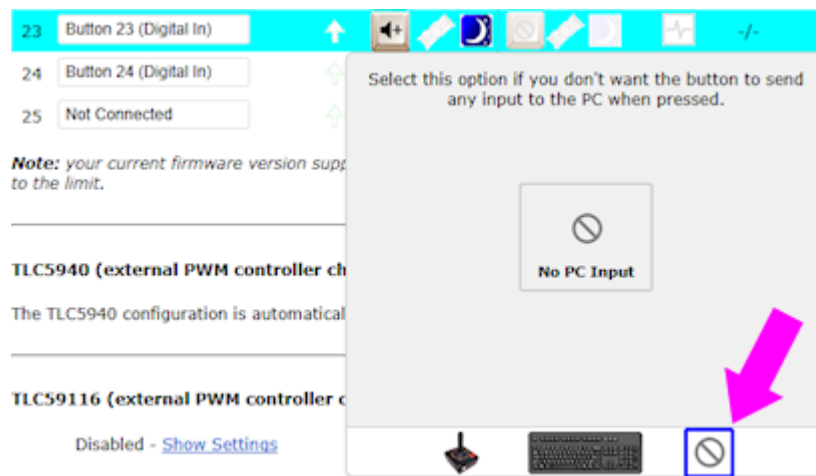




Also set the **Button type** option to match the physical type of switch you're using. If you installed a pushbutton, select "Momentary button"; if you installed a toggle switch of some kind that stays on until you turn it back off, select "On/off switch".

One last step: you'll probably want to make sure that the button you selected **doesn't** send any keystrokes to Windows when you press the button. To prevent it from sending any keys:

- Scroll to the **Button inputs** section
- Find the button you assigned (it'll be marked with a Night Mode icon)
- Click the key cap
- Click the "No Key" icon at the bottom



Of course, that assumes that you really don't want to send a key to the PC when you press the Night Mode button. In some cases, you might actually want to; for example, you might want to send a Media Mute key press, so that you mute the audio system at the same time you turn off the noisy devices. (If you're using PinVol, though, you can make it a little more nuanced than that, since PinVol has its own Night Mode that tracks the Pinscape Night Mode settings. See "PinVol coordination" below.)

Using a "shifted" button

If you want to be able to activate Night Mode via a button press, but you don't want to set up a separate dedicated button for it, you can use a "Shifted" button.

Pinscape lets you give two meanings to each button via a "Shift" button. The Pinscape "Shift" button is a little like the SHIFT key on a PC keyboard, in that it changes the meanings of all of the other buttons when you press it. But it's important to understand that it's not actually the *keyboard* SHIFT key. Pressing the designated Pinscape "Shift" button *doesn't* send a SHIFT key press to Windows. All it does is change the meanings of all of your other Pinscape buttons.

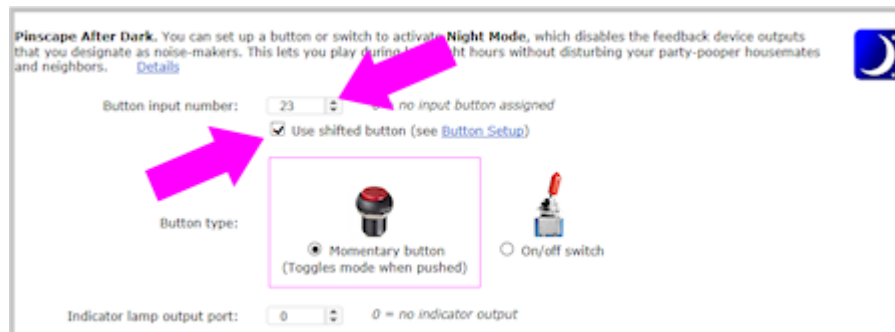
For example, I use the Extra Ball button on my cab as my Shift button. That works well for me because it doesn't get used very much in games (although it still works

as the ordinary Extra Ball button whenever that's needed). I use the Shift feature with my flipper and MagnaSave buttons to make them do double duty as audio volume controls. Under normal circumstances, they just act like normal flipper and MagnaSave buttons. But when I press and hold the Extra Ball button, they stop acting as flipper/MagnaSave buttons and start acting like Volume Up and Volume Down keys instead.

You can apply this same principle for the Night Mode button. For example, if you set up your Extra Ball button as the Shift button like I do, you could assign the Start button as the *shifted* Night Mode button. That won't take away your normal Start button - the Start button will act like it always has most of the time. But when you press Start while you're also holding down the Extra Ball button, Start gets its second, "shifted" meaning of activating Night Mode.

To set up a shifted Night Mode button:

- Set up a Shift button, as described in Chapter 110, Pinscape Button Inputs
- Select which button you want to use in shifted mode as the Night Mode button
- Open the Pinscape Config Tool
- Go to the Settings page
- Scroll down to the **Pinscape After Dark** section
- Enter the button number in the box, and check-mark the box saying "Use shifted button"



Night Mode indicator light

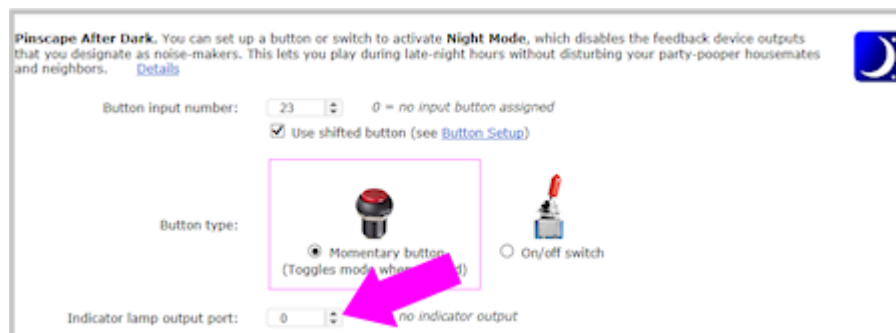
You can also set up an indicator light so that you can see at a glance whether or not Night Mode is engaged.

If you're setting up your Night Mode switch on a custom coin door service panel, you can make it a bit snazzier by using a lighted pushbutton, with the light in the pushbutton wired as the Night Mode indicator. Or you can just set up a separate LED somewhere to indicate the current mode.

Physically, the indicator light simply connects to any Pinscape feedback output port. I'd recommend using one of the "Small LED" outputs designed for the flipper button lights, since you'll probably only need a small 20mA LED for this indicator. The Small LED ports are limited to exactly this kind of small device, so you can't use them for much else, and you'll probably have a couple left over anyway. There are 16 of these ports total, and a full set of RGB flipper/MagnaSave lights only requires 12 of them.

Whichever port you decide to use, wire your chosen indicator light to it exactly as you would any other device. See Chapter 111, Pinscape Feedback Outputs for general port wiring instructions.

Once it's wired, open the Pinscape Config Tool, go to the Settings page, and scroll down to the **Pinscape After Dark** section. Enter the port number where you wired the light in the **Indicator lamp output port** box.



Note that you shouldn't use this port as a regular output port in your DOF setup, as the Night Mode indicator status will override whatever DOF tries to do with it.

PinVol coordination

If you're using PinVol to control volume levels, note that PinVol is automatically aware of the Pinscape Night Mode status, and will switch itself in and out of Night Mode to match the status on the controller. That saves you the trouble of setting up a separate key to control Night Mode for audio volume purposes.

Just like Pinscape uses Night Mode to turn off noisy devices, PinVol uses Night Mode to set a lower system-wide audio volume level. PinVol maintains a separate global volume level for Night Mode, so you can set up whatever reduced volume level you prefer when Night Mode is active, and then switch back to that same quieter level at any time by switching back to Night Mode.

114. TV ON Switch

The Pinscape Controller has a feature that lets you send a "Power On" command to your TV(s) when the overall system powers up. This feature can solve the problem that some TVs have when used in pin cabs, namely that they don't power on automatically when they're plugged in again after being unplugged.

This feature is only needed if your TV doesn't remember its power state when you plug it back in after it's unplugged, *and* you're using the sort of one-button power control setup as described in Chapter 11, Power Switching. That power control setup effectively unplugs the TVs and other devices from AC power when the PC is off, so each time you turn the PC back on, the TV thinks it's being plugged back in to the wall outlet after being unplugged. Some TVs automatically power up and return to displaying the last video source when this happens, but many just go into standby mode. The Pinscape TV ON feature is meant to solve this problem for TVs that go into standby mode, by explicitly sending a "Power On" command to the TV at system startup to get it back into operation.

Do I need it?

Before going on, you should make sure you actually need this feature. Some TVs need it and others don't. And regardless of TV, there's no need for it unless you're using one-button power control of the sort described in Chapter 11, Power Switching.

Assuming you're using that sort of power switching, here's how to test your TV to see if it needs help powering on at system startup:

- Set up the TV as a monitor on your PC, using the input on the TV (HDMI, etc) that you intend to use when it's in your cab. Turn everything ON and make sure it's all configured so that you see the Windows desktop on the TV.
- With the TV still ON, pull the plug out of the wall socket.
- Wait a few minutes.
- Plug it back in, and **don't push any buttons on the TV**.
- Did the TV turn back on and show the Windows desktop again?

If the answer is yes, you **don't** need the "TV ON" feature. A smart power strip is all you need to control this TV and get it to power up in sync with the PC.

If the TV stays off or goes into standby mode, you **do** need the TV ON feature. This TV won't turn on after being unplugged until it receives an explicit "Power On" command, which is exactly what the TV ON feature provides. You'll still need the switched outlet to control its line power, but you'll **also** need the TV ON feature to get it to wake up again each time power is restored.

Be sure to check each TV you plan to use in your cab, since some might need the TV ON feature even if others don't.

You can approximate this test without the PC if necessary. Just tune the TV to an over-the-air channel instead of using the PC as a video input. The important thing is that there's some kind of active video source, since some TVs will go into standby mode when there's no input and wake up when there is. The test with the PC as video source is the most reliable, because that's the exact setup you'll have in the cab, but most TVs will behave the same way with all video sources.

What if I'm still shopping for a TV?

Some cab builders try to avoid this whole issue by only buying TVs that pass the test above. So you might be wondering if there's an easy way to tell when shopping. Unfortunately, I've never seen a manufacturer or store mention it anywhere, and I don't think it would occur to most buyers to ask. It's not even safe to assume that similar models from the same manufacturer will behave the same way. If you want to be sure before you buy, you'll have to either find a store that's willing to let you test a floor model, or find a friend or forum member who has the exact model you're considering.

What this feature doesn't do

The TV ON feature **doesn't** control power at the plug, and it can't turn OFF your TVs. For both of those functions, use the techniques described in Chapter 11, Power Switching. The TV ON feature is meant to be used **in conjunction with** the switched power strip described in that chapter. TV ON isn't a replacement for the switched power strip; it's just an add-on to deal with TVs that don't automatically turn back on after being unplugged.

How it works

The TV ON feature works by having the Pinscape software send a "Power On" command to the TV a short time after the computer turns on. To physically send the command to the TV, there are two possibilities:

- By wires soldered to the switch terminals of the ON button on the TV itself. We connect these wires to a relay controlled by the Pinscape device. The Pinscape software pulses the relay shortly after the computer turns on. The TV registers this as an ON button press.
- By IR remote control. We connect an IR LED to the Pinscape unit, and position the LED so that it points at the TV's remote control sensor. The Pinscape software can be programmed to transmit the TV's remote control ON code at the proper time.

The Pinscape software sends the ON command a few seconds after the computer starts up. The delay is necessary because most TVs ignore commands for a few seconds after being plugged in. The exact amount of time needed varies by TV, so the delay can be adjusted in the Pinscape software configuration.

Using Windows commands

The normal way to use the TV ON feature is to let Pinscape do everything automatically using the **power sensing circuit** described below. If you're using the expansion boards, that's built in, so I'd go that route. If you're using a standalone KL25Z, though, the sensor circuit might be more trouble than you want to go to. So let me offer an easier alternative: use Windows commands.

Here's the idea:

- Create a .BAT file
- Put commands in the .BAT file to turn on the TV (see below)
- Put a shortcut to the .BAT file in your Start Menu > Startup folder

When Windows starts up, it'll run this .BAT file because it's in your Startup folder, so this will have roughly the same effect as the power sensor circuit. We just let Windows tell us when the system is starting up rather than detecting it electronically.

The commands all involve the utility program **PinscapeCmd.exe**, which you'll find in your Pinscape Config Tool folder. In the command examples below, we'll pretend this folder is named **C:\Pinscape**, but it doesn't have to use that exact name. Just substitute your folder path into the commands below if you're using something different.

If you're using the hard-wired TV ON switch, the command looks like this:

```
C:\Pinscape\PinscapeCmd TVON
```

The TVON sub-command briefly pulses the TV ON relay, just like the Pinscape software would do by itself at system startup when the power sensing circuit detects that the system has just been powered on.

If you're using an IR remote control instead of the hard-wired switch, do this:

```
C:\Pinscape\PinscapeCmd SendIR=1
```

That transmits the IR code in "slot 1" in your learned IR command list. If you're going to teach Pinscape multiple IR commands, change the "1" in **SendIR=1** to whichever slot contains the right ON command for your TV.

One more detail: if you have more than one Pinscape unit in your system at the same time, you have to tell PinscapeCmd which unit you want to send the command to. You do this by adding **Unit=2** (or whichever unit number you want to address) right after the **PinscapeCmd** command name:

```
C:\Pinscape\PinscapeCmd Unit=2 TVON
```

Power sensing circuit

The Pinscape expansion boards have some special circuitry built-in that detects when the system is powering up. This is a little more complicated than it sounds because of the way PC power supplies work. USB-powered devices like the KL25Z generally continue to receive 5V power even when the PC is off. That means that the Pinscape software can't tell from USB power alone when the system is powered on or off. As far as Pinscape's USB power goes, the power is always on, even when the overall system is off. Similarly, the Pinscape software can't assume that a KL25Z reboot means that the power status has changed, since the KL25Z can be rebooted at random times, such as when you update option settings, or if you should unplug it for some reason.

If you want to use Pinscape's automatic TV ON system **without** the expansion boards, you have two choices:

- You can make Windows run a batch script when the system boots, as described above ("Using Windows commands")
- You can build a replica of the power sensing circuit used on the expansion boards

The first option (a Windows batch script) is easier, but if you prefer to let Pinscape handle things automatically, read on for details on building the power sensing circuit.

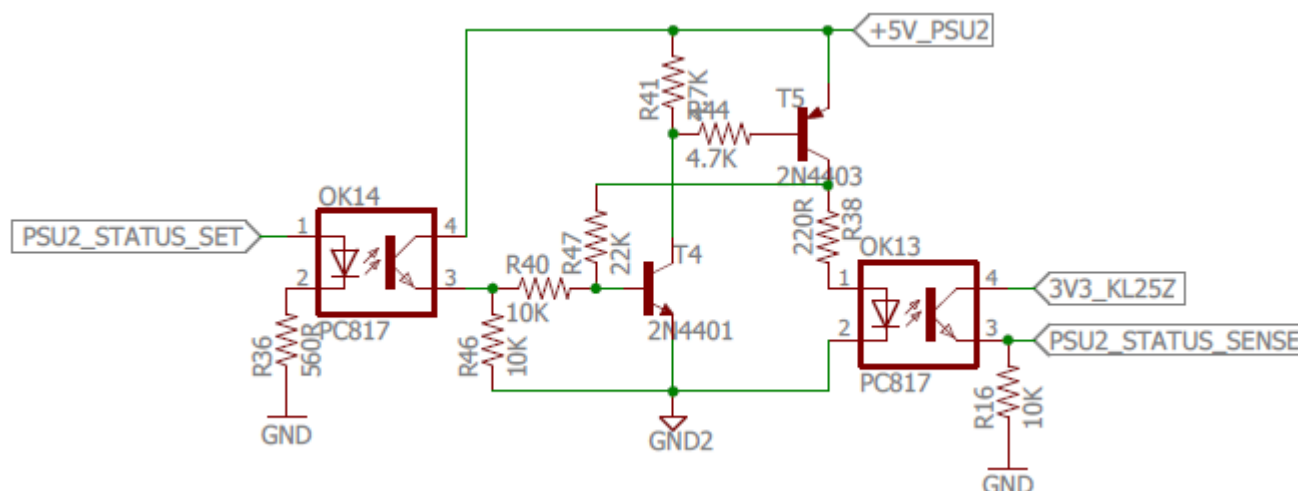
Note that this entire scheme depends on following the switched 120V power plan outlined in Chapter 11, Power Switching. In order for this to work, you must be using:

- A "soft" power switch for the main PC
- A switched outlet that's controlled by the PC power

- A secondary ATX power supply that's plugged into the switched outlet, for powering feedback devices and other systems apart from the PC motherboard

If you're already familiar with that design, read on. If not, you should go back and read about it in Chapter 11, Power Switching, because some of the material below won't make sense if you're not using the same design for the main power switching system.

Here's the schematic for the sensing circuit. Again, this is built in to the expansion boards, so you won't have to build this separately if you're using the expansion boards.



- Connect GND to a Ground pin on the KL25Z (see Appendix 1, KL25Z Pin Out)
- Connect 3V3_KL25Z to a 3.3V pin on the KL25Z (see Appendix 1, KL25Z Pin Out)
- Connect GND2 to the **secondary** power supply ground (the one plugged into the switched outlet)
- Connect +5V_PSU2 to the **secondary** power supply +5V
- Connect PSU2_STATUS_SENSE to the KL25Z GPIO port assigned as "Power status input" in the Pinscape Config Tool settings
- Connect PSU2_STATUS_SET to the KL25Z GPIO port assigned as "Status latch output" in the settings

You can use any free GPIO ports for the connections listed. The expansion boards use PTD2 for the power status input (PSU2_STATUS_SENSE) and PTE0 for the latch output (PSU2_STATUS_SET).

Configuring settings for the power sensing circuit

To configure your standalone TV ON circuit, run the Pinscape Config Tool. Go to the Settings page, and scroll down to the **TV ON Switch** section.

TV ON switch. If one or more of your monitors needs to be turned on manually every time you power up your cabinet, you can use this feature to switch them on automatically. See the Build Guide for wiring instructions.



☒ Enabled

[Test](#)

Power status input:

PTD2

Status latch output:

PTE0

Relay output:

PTD3

Startup delay time (seconds):

7

This is how long to wait after power-on before pulsing the relay

Click on the **Power status input** box to select the GPIO pin you wired to PSU2_STATUS_SENSE. Then click on the **Power latch output** box to select the pin wired to PSU2_STATUS_SET.

If you're using the IR transmitter instead of the hard-wired TV ON relay switch, you can simply set the **Relay output** box to **Not Connected**, to tell the software that no pin is connected to that function.

The **Startup delay time** is the amount of time in seconds that the software will wait after system startup to send the TV ON signals to the TVs. This is used for both the relay switch and the IR signals. Adjust this as needed to make sure that the system waits long enough for your TVs to initialize. Most TVs aren't responsive to any control inputs for a few seconds after they're plugged in, so this delay is intended to give the TVs long enough that they're ready to handle the ON signal when Pinscape sends it. If your TVs usually work but seem to miss the signal some of the time, adding a few seconds to the delay might make it more reliable.


When you've entered all of the settings, click "Program KL25Z" at the bottom of the page to save the changes to the device.

Setting up an IR transmitter/receiver

If you're going the IR route, the minimum requirement is an IR transmitter, to send IR commands to your TV(s). But you'll almost certainly want to set up an IR receiver as well, so that Pinscape can "learn" commands from your remotes. It's possible in principle to program the remote control codes by hand, without the learning function, but it's surprisingly difficult to find published information for the exact model of TV you're trying to control. And even if you can find published codes, they're hard to make sense of because no one can agree on a universal standard format. The learning function is by far the easiest way to figure out what codes your exact TV is using, because it lets Pinscape read the codes directly from your remote control.

See Chapter 112, IR Remote Control for instructions on setting up the IR remote control features with Pinscape.

Once you've installed the IR transmitter and receiver hardware, you can set up the TV ON feature through IR as follows:

- Teach Pinscape the remote control code for your TV's "ON" button, following the procedure in Chapter 112, IR Remote Control ("Learning IR commands from your remotes")
- Click the TV ON icon () in the row for the "ON" button code
- Be sure to save settings by clicking "Program KL25Z"

Once you activate the TV ON icon for a code, Pinscape will automatically transmit that code at system startup, when the power sensing circuit detects that power has just been turned on. Remember that you **must** use and configure the power-sensing circuit for this to work (the circuit is built in to the expansion boards, but you have to build it separately if you're using a standalone KL25Z).

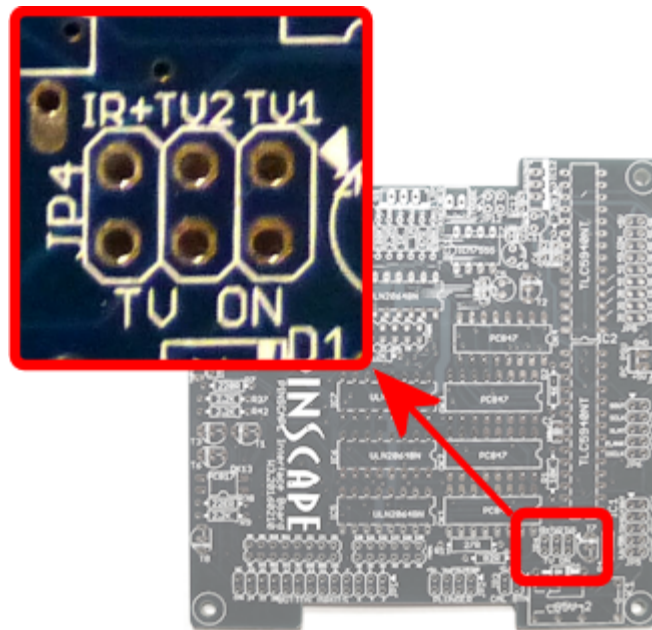
Connecting a hard-wired TV ON switch

Warning: I no longer recommend this approach. Even though it's the original solution I used in my own cab, I'd recommend avoiding it and using the IR transmitter solution instead. Hard-wiring the power switch requires opening up the TV, and modern flat-panel TVs are just too delicate for this to be a good idea. The IR solution is completely non-invasive.

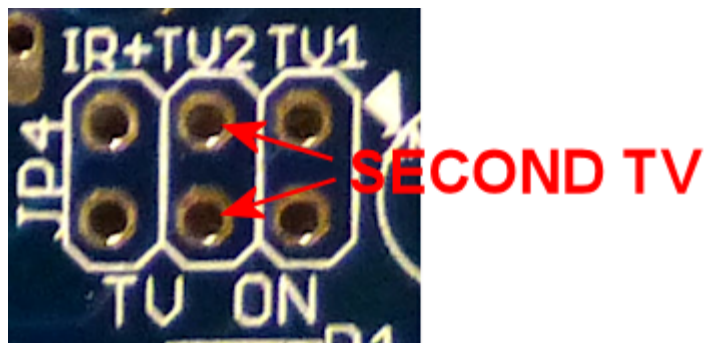
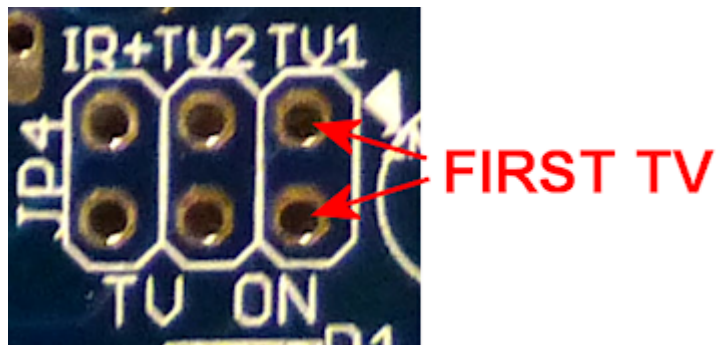
If you're absolutely dead set on this approach, though, read on.

Expansion boards

The main expansion board has wiring for up to two TV switches. The connectors are on the JP4 pin header, labeled "TV ON".



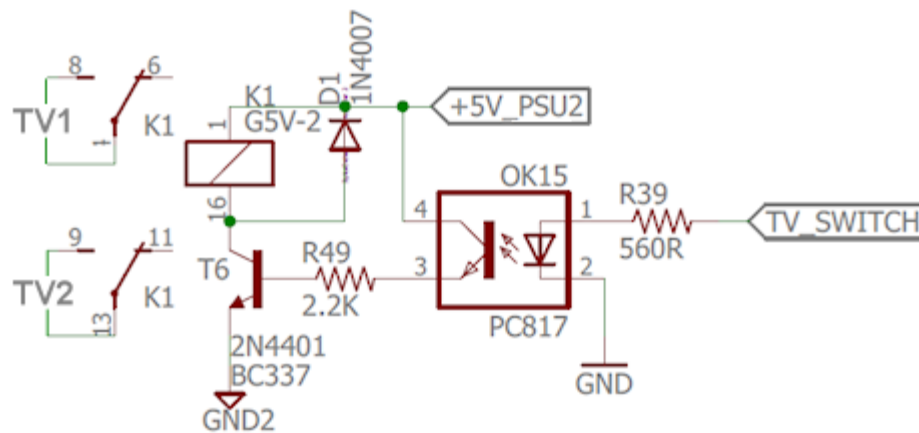
Connect the first TV's power switch to the pair of pins in the "column" under the label TV1. Connect the second TV's power switch to the pair of pins in the column under the label TV2.



You can just run ordinary wire between the expansion board pins and the switch on the TV. These carry only low-power switch signals, so you can use a fairly thin wire like you'd use for wiring cabinet buttons, such as 24 AWG.

Standalone wiring (no expansion boards)

If you're using the expansion boards, the relay switch is built in. If you're not using the expansion board, you can build the same circuit yourself. Here's the schematic:



- Connect GND to a Ground pin on the KL25Z (see Appendix 1, KL25Z Pin Out)
- Connect +5V_PSU2 to the +5V of your secondary power supply
- Connect GND2 to the ground (black wire) of your secondary power supply
- Connect TV_SWITCH to the GPIO pin you're using for the TV relay

You can use any GPIO pin to control the TV relay. The default used on the expansion boards is PTD3, but you can use a different pin if that one is in use. You'll have to configure whichever pin you select in the Pinscape Config Tool. In the Config Tool, go to the Settings page, then scroll down to the TV ON section. Click the **Relay output** box to set the appropriate GPIO pin.

Wiring the TV switch

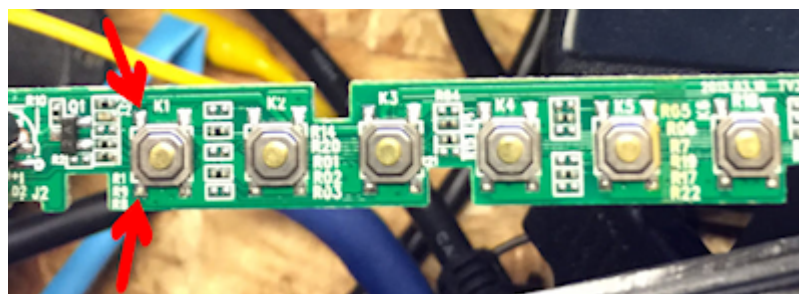
You'll have to be comfortable with taking the TV apart at this stage, because we have to connect some wires to the On/Off button.

There are no generic instructions for taking a TV case apart, so you're on your own for this part. Your goal is to open the case and expose the little circuit board containing the On/Off button.



Needless to say, use extreme caution with this step. In modern LCD TVs, the LCD panel and polarizing filter are very thin, brittle plastic sheets and often have no structural support other than the outer case, so it's very easy to crack them during the removal process or after the case is off. Removing the case will also void the warranty, so you're assuming the entire risk of breaking something by proceeding.

Once you get the case open, you should find a little circuit board located under the area where the buttons on the case are situated. It's usually long and narrow, and looks something like this:

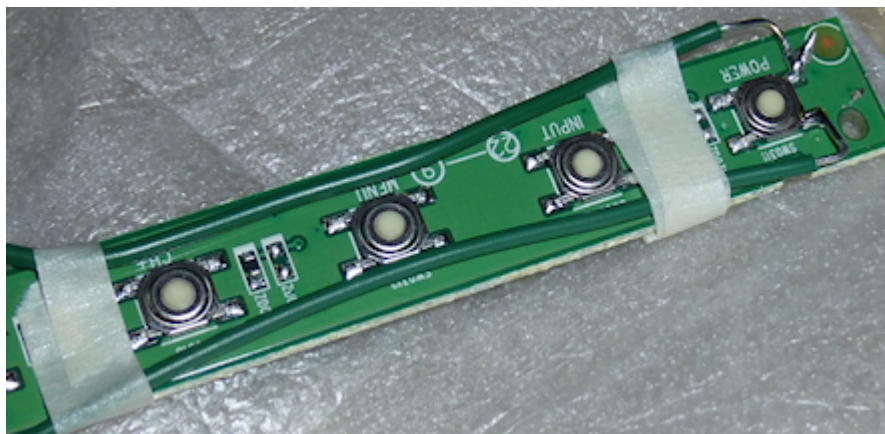


The red arrows in the photo above show the soldering points for the button leads. The little squarish silver objects are the buttons. These are normally situated immediately under the exterior plastic buttons on the TV's bezel; pressing on the exterior plastic button has the effect of pushing down on this metal part, which is the real button.

Once you find this circuit board, identify which button corresponds to the On/Off button on the outer case. Do this by position: just find the inner button that's situated underneath the On/Off button on the case. You can also do this by counting buttons from right to left, since there should be the same number of silver buttons on the circuit board as plastic buttons on the case.

Next, identify the switch leads. There are probably four leads to these switches, one at each corner. On the TVs I've looked at, the leads are in pairs that are electrically connected together, so there are really only two wires here even though it looks like four. Put your multimeter in continuity test mode and check the leads in pairs. Find a pair that are **not** connected normally, but that become connected when you press the button. These are the leads you want to solder to.

The next step is possibly even more delicate and tricky than opening the case. You have to solder wires to the button leads you just identified. To do this, use fine hookup wire, 24 AWG or thinner. Strip a very short length of insulation from the ends, around 1/8". Melt a little solder onto the end of the wire. Position the end of the wire at the desired contact point. Now get out some tape (I used thin strips of masking tape here) and secure the wire to the board a couple of inches away from the contact point. The idea is to hold it in place at the desired position before soldering so that the solder can just flow over the junction with everything already positioned properly. Once everything is in place, heat the end of the wire for a few moments, long enough for the solder to melt and flow onto the switch lead. Remove the soldering iron carefully and try to hold everything very still for a few moments so that the solder can solidify over the junction point. If all went well, the wire should stick to the switch lead. The connection will be delicate at best, so you'll want to secure the wire with a couple more pieces of tape to minimize mechanical stress on it.



TV On/Off switch with wires soldered to leads

Repeat this process for the second lead. Once both are soldered and held securely in place with tape, test your work with the multimeter. Use continuity test again. Connect the meter leads to the free ends of the wires you just soldered. The meter should read open/no connection. Press the button, and the meter should read closed/connected. If that works, you're set. Put the TV case back together, taking care to run your newly attached wires out a suitable opening.

Now just connect the two wires from the TV's switch to the appropriate TV ON pin pair on JP4 on the expansion board (see the illustrations above). The order of the wires doesn't matter: you can connect either switch terminal to either pin in the TV ON pair.

115. Troubleshooting

Here are some miscellaneous tips for troubleshooting, based on problems that people have reported in the forums.

Quick index:

- Button #6 isn't working
- USB disconnects
- Solenoids/flashers fire briefly when powering on the system
- The TV ON feature doesn't turn the TV on at system startup
- Ball bounces around on the plunger in VP
- ZB Launch Ball launches by itself
- ZB Launch Ball doesn't work at all
- Buttons/axes are missing in joystick control panel
- Flipper buttons are flaky
- Nudging doesn't work in VP
- DOF effects or sound effects make the ball fly around in VP
- Chapter 46, General DOF troubleshooting

Button input #6 isn't working

Symptom: all of your other buttons are working, but button #6 won't generate any key presses on the PC, no matter which key you map it to.

The usual reason this happens is that you're using the Expansion Boards, but your KL25Z is still configured with the default pin settings. By default, the firmware is configured for the standalone KL25Z setup, with no expansion boards. The button wiring is *almost* identical between the expansion boards and the standalone setup. But it differs in one place: button #6 is wired to a different GPIO pin on the expansion boards. (This wasn't an arbitrary change just to make your life more difficult; the GPIO port that button #6 uses in the standalone setup is needed for another purpose in the expansion boards.)

The quick solution is to go to the top of the Settings page in the Config Tool, where you set the system type. If it's currently set to Standalone KL25Z, change it to Expansion Boards. While you're at it, make sure to set the correct number of each type of board you're using. Click "Program KL25Z" at the bottom and let the device reset.

If that doesn't fix it, you might have a wiring problem or some other configuration issue. For help, see:

- "Button inputs" in Chapter 97, Debugging the Expansion Boards
- Chapter 110, Pinscape Button Inputs

USB devices disconnect or reset

Symptom: Pinscape or other USB devices disconnect or reboot themselves when they shouldn't.

For example, you're in the middle of playing a game in VP, and you hear that

Windows bleep-bloop tone meaning that a USB device has been unplugged, or your button inputs just stop working, or your plunger stops working, or DOF output stops working.

The right approach to debugging this depends on how it's happening, so read on for the common scenarios.

It happens completely at random

"Completely at random" means that you can get the disconnects to happen when nothing at all is going on. You're not touching the computer, and there's no software loaded, just the Windows desktop.

If you *feel* like the problem is random, but you're always doing something with the computer when it happens, you should actually try that idle scenario we just described to find out if your hunch is correct. Just leave the computer running idle for a while and see if you observe any disconnects. If you can leave it running idle for a long time *without* any disconnects, then the disconnect is probably being triggered by some specific cause. In that case you should try to figure out what that trigger is. The most common triggers have to do with DOF devices - shaker motors, replay knockers, solenoids, etc. See below for more on that.

But let's say that it *is* completely random: you can indeed observe Pinscape disconnecting sporadically when nothing at all is happening on the computer. The most likely problem in this case is a USB compatibility problem between the KL25Z and your motherboard. These sorts of problems are extremely difficult to diagnose with any analytical certainty, so we have to resort to what I call "shotgun debugging", where you throw a bunch of fixes at the problem and see if anything helps:

- Try plugging the KL25Z in through a USB hub instead of plugging it directly into your computer. The best bet seems to be a powered USB 2 hub. This seems to be by far the #1 fix for compatibility issues, so if this doesn't help, the problem might not be a compatibility conflict after all.
- Make sure your Windows USB device drivers are up-to-date. Open Device Manager and go through the list of Universal Serial Bus controllers, particularly the "Generic USB Hub" and "USB Root Hub" devices. Try updating drivers via Properties > Driver tab > "Update driver".
- You might also just have something flaky somewhere in the system, like a loose USB cable or a bad solder connection. Check USB cables and any soldered wire connections you've set up, just to make sure you can't spot anything that looks suspicious.

It happens every time my shaker motor/replay knocker/something else fires

Symptom: Pinscape (or another USB device) disconnects from USB every time your tactile feedback toys activate while playing a game in VP. For example, this might happen when your shaker motor runs. It might happen reliably or sporadically.

The most likely cause is electrical interference, caused by voltage spikes from those high-power solenoid or motor devices. Computers and logic devices like Pinscape are sensitive to tiny changes in voltage, so it's inherently difficult to combine logic devices and high-power solenoid devices in a single system the way we do in a pin cab. But there are some well-established ways of "filtering" the electrical noise so that the logic circuits won't go haywire.

- **Diodes:** The #1 cause of electrical interference in virtual pin cabs is the

"flyback" voltage that coils and motors produce when they switch off. So the first thing to do is make absolutely sure that you have diodes properly installed in all of your inductive devices: motors, solenoids, contactors, relays, replay knockers, chime coils, bells. You need diodes for everything like this, even small relays used for switching. Read Chapter 53, Coil Diodes if you haven't already installed diodes on everything that needs them. If you have already installed them, it might be worth checking that they're all connected properly and that none of them have come loose. It's best if they're permanently soldered into the wiring so that you can't accidentally disconnect them.

- **Separate power supplies:** It's best to not to connect any feedback devices to your PC power supply. Use a separate power supply for feedback devices instead.
- **Make sure power supply grounds are interconnected:** See Chapter 45, Power Supplies for Feedback for details. Double-check that the ground connections between the power supplies are all solid - check for any wires that might have come loose or might not be firmly connected.
- **Add a small capacitor (for solenoids):** If the problem seems to be coming from a particular solenoid, relay, contactor, or other solenoid-like device, and you're sure the diode is installed correctly and solidly connected, try adding a small disc capacitor, perhaps 0.1 μ F. Use a disc capacitor, not electrolytic; these are unpolarized, so you don't have to worry about which direction it goes. Connect it exactly the same way as the diode, across the input leads to the solenoid. Leave the diode in place as well, of course. (So the diode and capacitor will be "in parallel" with each other, if you know that electrical term.)
- **Add a motor EMI filter:** Motors generate their own unique kind of electrical interference in addition to the flyback voltage. This isn't commonly needed in a virtual pin cab, but it might be worth a try if nothing else helps: add a pair of small inductors, also known as chokes, in series with the motor. Place the first one in series with the positive voltage power connection to the motor, and the second in series with the connection to the feedback device controller. Try 4.7 μ H inductors, with an amperage rating higher than the motor's actual amperage. (μ H stands for micro-Henry, a measure of inductance.) Here's an example:

Coilcraft DR0608-472L 4.7 μ H, 5.8A radial inductor - at Mouser

See "Electrical Interference" in Chapter 61, Shaker motors for a sample wiring diagram.

- **Get a bigger power supply:** This is also uncommon, but the problem might be that you're "browning out" your whole system by overloading the feedback device power supply. This can cause the voltage levels to fluctuate, which can manifest as USB disconnects or other computer problems. A larger power supply might help by keeping the voltage level more stable. "Larger" means a higher total wattage rating.
- **Try a different power supply:** It might also help to simply try a *different* power supply for your feedback devices. Power supplies have their own power line filtering internally, and some of these work better than others. It might be that your feedback power supply just isn't blocking enough of the electrical noise from your solenoids and motors, and the noise is finding its way into your PC circuitry through the common power wiring.

It seems to happen when motors/solenoids fire, but not every the time

Symptom: The disconnects seem to happen when you're in the middle of game in VP

and there's lots of action with the DOF toys, like when a bunch of bumper contactors fire in a short period.

This is almost certainly the same sort of problem as "It happens every time..." above. Try the same fixes listed there. These sorts of problems are actually more likely to be sporadic than to happen every time, so a degree of randomness doesn't change the likely causes.

The TV ON feature doesn't turn the TV on at system startup

Symptom: The TV isn't turning on at system startup, even though you've programmed the TV ON feature to send IR commands to the TV, or connected a relay to the TV's ON button.

The first thing to check is that the IR commands or TV relay are working at all. Both can be exercised via the Pinscape Config Tool:

- For the relay switch, use the TV Relay Tester dialog, which can be accessed from the Config Tool's main screen
- For the IR remote transmitter, go the Settings page, scroll down to the IR Remote Control section, and use the "test" button next to the command code you want to try sending

If the relay and/or IR transmitter are working in test mode, the next thing to test is that the "power detection" circuit is working correctly. If you're using the expansion boards, the power detection circuit is built in. But - and this is important - if you're using a standalone KL25Z, you have to build the power detection circuit separately. See "Power sensing circuit" in Chapter 114, TV ON Switch for the circuit plans. The TV ON system depends upon this extra circuit, so if you didn't build one, that's probably why the TV on feature isn't working.

To test the power detection circuit, you have to watch the KL25Z's on-board diagnostic LED while powering your system on. It should show **slow blue blinking** during the TV ON delay period - typically about five seconds - immediately after you power on the system. The delay time is programmable via the Config Tool, so you should see the slow blue blinking for the amount of time you programmed. If the LED doesn't show the slow blue blinking, the power sensing circuit probably isn't working correctly. Check the wiring and the configuration settings for the TV ON section in the Config Tool.

The final suggestion I have, if everything above checks out, is to try a longer delay period in the TV ON settings. The purpose of the delay period is to give your TV time to "reboot" after the 120V power is connected. Remember that we've set things up so that the TV is effectively unplugged when your pin cab is powered down; when you turn the pin cab on, it's like plugging the TV back into the wall socket. Most modern TVs need several seconds to gather their wits when you plug them in. They usually don't respond to any IR commands or button presses during this time. That's what the programmable TV ON delay period is all about: it's to give your TV time to boot up and start listening for IR commands and button presses. The exact time required varies by model, so what works for me might not work for you. So you should try a longer delay time to see if your TV just needs more time to become responsive after power is connected.

Solenoids/flashers fire briefly when powering on the system

Symptom: When you turn on power to the system, some of the feedback devices

energize briefly. For example, the flashes all flash white briefly, or you hear contactors or solenoids fire for just a moment. This might happen sporadically or consistently.

Cause: In all likelihood, especially if it's sporadic (that is, it only happens on a fraction of power-ups, at random), it's just a design limitation in the controller. (Or perhaps a design flaw, depending on how you look at it.) Some output controllers power up with the output channels in a random state, so some ports might be activated when you first turn on the power. This should only be momentary, because the controller's software should deactivate all ports as soon as it starts up, but the software usually takes a few moments to get going after the power comes on. The random activation happens in this brief window. The Pinscape expansion boards and LedWiz both exhibit this behavior.

Solution: My "solution" is to just ignore it. As long as the misfire events are only momentary, they're not going to damage anything. It can be a little alarming if a bunch of noise-making devices like solenoids all fire at once, but other than rattling your nerves, a momentary activation won't damage the devices; they're built to fire repeatedly and frequently during normal play, after all.

The ideal way to fix it would be to change the design of the controller device to eliminate the random startup state. That's obviously not feasible with a commercial device, and unfortunately I wasn't able to find a way to address it in the Pinscape boards. The random startup state there comes from the PWM chips we use, and working around it would have required added circuitry, which I couldn't find room for.

An alternative fix that *is* feasible to pursue, if the glitch bothers you enough, is to add a delay timer to the power supplies for the feedback devices. The idea is to prevent the feedback device power supplies from powering up until after the controller has finished initializing. Without power, the flashers and solenoids won't be able to fire, no matter what the state of the output controller ports. You can find multi-function relay timers on eBay that can do this. With a power-on delay timer, you could wire the mains (120V) power to the feedback power supplies through the timer relay, so that the feedback devices don't receive any power until after the controller is fully initialized. A few seconds should be sufficient.

Ball bounces around on plunger in VP

Symptom: the ball bounces around wildly in VP when it's sitting in the plunger lane, even when I'm not touching the plunger, and maybe even launches itself. It stops when I pull back the plunger.

This is usually a problem with plunger calibration.

- First, make sure that you **don't** have any Windows joystick control panel calibration in effect. Everyone tries the Windows joystick calibration process because it just sounds like something you should do, right? But it's actually designed for joysticks, *real joysticks*, the kind with a stick and some buttons on top, and Pinscape isn't one of those. It only pretends to be a joystick for the sake of the software interface. The Windows calibration process wreaks havoc with Pinscape and makes all of its input wildly random and wildly wrong. If you've ever run it, it will make your nudge and plunger inputs act erratically. One of the common symptoms is that the ball bounces around in the plunger chute; another is that the nudge input is crazy.

Fortunately, it's really easy to undo the damage from past calibration attempts. Even if you're not sure that you've ever run calibration, do this:

- Press Windows+R, type **joy.cpl**, press Enter
- Find **Pinscape Controller** in the list and double-click
- Click the **Settings** tab
- Click **Reset to defaults**
- Now make sure that you've gone through the **Pinscape Config Tool** plunger calibration process. This is a whole separate calibration scheme from the Windows joystick calibration - the Windows scheme is for real joysticks, and this one is specifically for plungers.
 - Run the Pinscape Config Tool
 - Find the device and click the Plunger icon
 - Click the Calibrate button
 - Follow the on-screen steps to perform the calibration

Note that you shouldn't have to repeat the calibration process as long as you don't mess with the physical plunger setup. If you make any physical adjustments to the sensor, or you reinstall the plunger itself for some reason, you should repeat the calibration. You can also repeat it any time it seems out of whack, as sensors can change electrically over time, but it's not something you should have to do with any frequency.

- If the ball is still bouncing around, and you can also see the plunger itself jumping around erratically (while you're not touching anything), you might have to add some "jitter filtering". Go back to the Pinscape Config Tool and click the plunger icon again. Without touching the plunger, is the green bar in the sensor viewer dancing around visibly? If so, try increasing the "jitter filter" number until the random motion stabilizes.

The jitter filter is there to smooth out the random motion that can come from analog sensors like potentiometers and IR distance sensors. Analog sensors tend to have a little bit of variation from one reading to the next, even when the plunger is perfectly still, because a digital reading from an analog sensor is always an approximation. Each approximation tends to be a little different from the previous one. That shows up as random motion, which I call "jitter". The jitter filter smooths that out by ignoring small variations in readings - exactly how small is determined by the "window size" you specify. A larger window smooths out larger variations - but at the expense of less accurate readings. You want the window to be as small as possible, just enough to smooth out the visible random motion in VP.

ZB Launch Ball launches the ball by itself

Symptom: In games where ZB Launch Ball is used, the ball keeps launching by itself, before I do anything with the plunger.

This is usually caused by the same problem as "Ball bounces around on plunger" above. The ZB Launch Ball feature works by detecting when the plunger moves in front of the resting position. If the plunger isn't calibrated properly or has too much random sensor jitter, Pinscape can get false readings that it interprets as the kind of forward push that activates the ball launch. Try the fixes for the bouncing ball problem above.

Also refer to "Troubleshooting" in Chapter 109, ZB Launch Ball.

ZB Launch Ball isn't working at all

Symptom: The ZB Launch Ball feature is enabled, but it won't work. The ball won't launch in plunger-less games.

The ZB Launch Ball feature only works when DOF activates it, which means that the table you're running in Visual Pinball has to be configured properly in DOF.

See "Troubleshooting" in Chapter 109, ZB Launch Ball for steps to try.

Buttons/axes are missing in the Windows joystick control panel

Symptom: When you go to the Windows joystick control panel ("Set up USB Game Controllers") and look at the Pinscape device, it looks weird. For example, it doesn't show all of the X, Y, and Z joystick axes, or it doesn't show 32 buttons.

This is caused by corrupted device information in the Windows registry, which can happen if there's a problem during the initial USB connection setup when you plug in the Pinscape device. The annoying thing is that Windows caches the corrupted information, so if you have a connection problem at any point, it can leave cruft behind that keeps showing up even when the connection later succeeds.

To fix this, you have to delete the corrupted registry key with RegEdit. You might need to run RegEdit in Administrator mode to do this - if RegEdit shows a permissions error when you try to delete the key, or if the key just won't go away permanently when you delete it, try exiting RegEdit and running it again by right-clicking RegEdit and selecting "Run as Administrator" from the menu.

Here's the procedure:

- Disconnect all of your Pinscape devices
- Open RegEdit
- Navigate to this key:

HKEY_CURRENT_USER\System\CurrentControlSet\Control\MediaProperties\PrivateProperties\DI

- Find all of the sub-keys that look like one of these ("xxxx" can be any sequence of four letters or digits):

VID_FAFA&PID_xxxx

VID_1209&PID_EAEA

- Open each matching key, and delete its **Calibration** subkey
- Exit RegEdit

Plug the Pinscape device back in and check again in the joystick control panel. It will hopefully show the correct controls now. If not, try the whole process again with "Run as Administrator" (if you didn't already). If even that doesn't work, try the whole thing again, and reboot the computer after exiting RegEdit. Rebooting really shouldn't be necessary, but sometimes things stick in Windows caches until you do, so try it as a last resort.

Here's a little background information, in case you're wondering what those VID/PID keys are and why you're looking for these particular ones.

The **VID_xxxx&PID_xxxx** keys are tied to the device's USB ID, which is something you can select in the Config Tool. Most people leave it with the default setting, which

uses an LedWiz-compatible USB ID, which appears in the registry with the **VID_FAFA** prefix mentioned above. So that's the most common thing to look for. You'll only see the **VID_1209** key if you intentionally changed the USB ID to use the non-LedWiz "Pinscape" USB code instead. (The Config Tool also allows you to select a completely custom code, but that's not something you'd do in normal use - it's really only for people who want to repurpose the firmware code for something other than virtual pinball. But on the off chance that you are using a completely custom USB ID, then you'd have to look for a VID_xxxx&PID_xxxx key matching your custom ID, instead of one of the standard ones listed above.)

You'll probably see a bunch of other VID_xxxx&PID_xxxx entries in your registry as well. Those are for other, unrelated devices, such as other joysticks or gamepads. You can ignore those for the purposes of this procedure.

The **Calibration** subkeys that we're deleting come from the DirectInput subsystem, which caches a bunch of information about gaming devices when they're first plugged in. This information comes from the device itself, so basically, Windows is asking the device about itself and then storing (in the registry) its own copy of the information. The point is to avoid having to repeat the data exchange every time you reboot Windows, to speed up reboots. It's fine when everything works perfectly, but it can cause problems if the initial information exchange has any sort of USB communications glitch. If anything gets garbled in the initial exchange, Windows just goes ahead and stores the garbled data, and never bothers to check again to see if a correction is necessary. I personally think this is a bad design on Microsoft's part, since USB errors are common enough that the initial exchange actually does get garbled once in a while. The "wrong number of buttons" or "wrong number of axes" problem is exactly how this manifests - Windows ran into an error when first interrogating the device, and then made the erroneous information permanent by storing it in the registry. The procedure we're doing here simply deletes the cached information and forces Windows to interrogate the device again the next time you plug it in. With luck, the data exchange process won't run into any errors on the new attempt, and Windows will store the correct information.

Flipper buttons are flaky

Symptom: Your leaf-switch flipper buttons don't work reliably. You might see this as random auto-repeat keys on the PC, extra keystrokes while you're holding down the buttons, or other intermittent behavior.

You should start by checking your other cabinet buttons to make sure they're not also exhibiting similar behavior. In particular, compare the behavior to any buttons you have that use microswitches rather than leaf switches. (Microswitches are the little plastic boxes with the switch assembly fully enclosed inside. The standard pinball "Start" buttons use this kind of switch.)

If your microswitch buttons are also acting flaky, the problem is probably with either your key encoder or with the wiring between the switches and the key encoder. Check the wiring, particularly the "common" or "ground" connection that all of the buttons share. Also check your key encoder's instructions to make sure that you've wired it correctly and that you've done any necessary software setup for it on the PC.

If it's *only* the leaf switches that are acting flaky, I'd still start by double-checking the wiring to make sure it's solid. Assuming the wiring looks good, there are a few things you can try.

First, make sure you have the right kind of switches. There are actually two kinds of leaf switches, for different purposes, and it's important to have the right type in a pin

cab. Some leaf switches are designed for high-voltage power connections, and some are designed for low-voltage data connections. A pin cab requires the low-voltage type. If you bought your leaf switches from a pinball vendor like Pinball Life or Marco Specialties, and they were sold specifically as **flipper button** switches, they might well be the high-voltage type, because many of the real pinball machines that use leaf switches for the flipper buttons are wired so that the switches directly control the 50V flipper coils. Those high-voltage switches have contact points made of tungsten, because it's tough and durable enough to withstand the high voltages. In contrast, low-voltage leaf switches have gold-plated contact points. Gold is a better conductor than tungsten, which is why it's better for a low-voltage data switch, and why it's the type needed for a pin cab. Tungsten contact points aren't conductive enough for reliable low-voltage switching, so they can make the buttons flaky when used in a pin cab. Tungsten also oxidizes over time, which further reduces its conductivity, so tungsten switches might work fine at first but start acting up after they've been deployed a while. This can sometimes explain situations where your switch problems only appeared recently or seem to be getting gradually worse.

You should be able to tell which type of leaf switches you have by visual inspection. Take a close look at the little disks at the ends of the switch leaves - those are the contact points. Tungsten contact points look dull and dark in color, whereas gold-plated contacts are shiny and light-colored. If you think you have the high-voltage tungsten type, you should try replacing them with the low-voltage gold type. The last I checked, VirtuaPin only sells the gold type, so you're probably safe if you bought your switches there. But Marco Specialties, Pinball Life, and all of the other pinball parts vendors sell both types, so you have to be careful when ordering to buy the low-voltage, gold-plated type.

Second, you might need to adjust the gaps between the contacts. Leaf switches are notoriously finicky this way, and even brand new ones might need to be adjusted when first installed.

See "Adjusting the switch gap" in Chapter 26, Inside the Cabinet for instructions.

Third, you might simply try cleaning the contact points. This is necessary from time to time in real pinball machines because of the copious grime generated by all of the mechanical action, but it shouldn't be as much of an issue in a virtual pin cab. If you do suspect that dirty contacts are making the switches flaky, try gently cleaning the contact points with a Q-tip dipped in rubbing alcohol. **Don't** use anything abrasive and don't clean too aggressively. The gold on the low-voltage contacts is an extremely thin plating layer that can easily be removed by abrasion.

Nudging doesn't work in VP

Symptom: nudging isn't working at all in Visual Pinball. There's no effect on the ball when I give the cabinet a good nudge.

This is mostly likely a problem in the VP configuration.

- Go back through the VP nudge setup procedure in Chapter 36, Nudge & Tilt
- Make sure the axis settings in the VP dialog match your device's axis settings. Most nudge devices use the X and Y axes by default for nudging, but double-check that in your device setup. The Pinscape Config Tool lets you switch to the Rx and Ry (rotational) axes instead, so if you made that change in the Pinscape setup, you'll have to make the same change in the VP setup. The two setups don't have any way to talk to each other on their own.
- If you have other joystick devices in your system (actual joysticks, or other

devices like Pinscape that *pretend* to be joysticks), try removing all of the other ones. VP isn't very good at handling multiple joysticks. Even if you don't want to remove the others permanently, at least try this as a test to see if it fixes the problem. If that fixes it, there are a few possible approaches to dealing with the conflict while keeping the other devices in your system:

- Check the other devices to see if you can disable their joystick functions while keeping their other capabilities.
- Check the other devices to see if you can change the joystick axes they're using. In particular, see if you can get them to stop sending any data on the X and Y axes. You might be able to tell the other devices to use the rotational Rx and Ry axes, for example.
- If you're using Pinscape, use the Pinscape Config Tool to change the Pinscape accelerometer to use the Rx and Ry rotation axes. Make the same change in the VP setup.
- Microsoft has a tool called Device Console, or **DevCon**, that can selectively enable and disable individual devices from a batch script. Some people have resolved conflicts by creating a batch script that disables conflicting devices just before each VP launch, and re-enables the devices after VP exits. You can find DevCon in the Microsoft Windows Driver Kit.

Vibrations or sound effects make the ball fly around in VP

Symptom: The ball flies around or veers off course in Visual Pinball whenever a solenoid fires, or when the shaker motor fires, or when a loud noise comes through the subwoofer.

The problem is probably that your accelerometer is picking up the vibration from the feedback devices or speakers, and VP is reading it as "nudge" input. There are two reasons this could be happening:

- The first, and by far most common, is that your accelerometer settings in VP are configured to be much too sensitive. The solution is to reduce the "gain" settings until the vibration stops affecting the simulated ball motion.

Almost everyone initially sets the accelerometer gain in VP to a setting that's way too high, based on a natural desire to see your new toy in action. Specifically, everyone wants to see the ball react in VP, clearly and conspicuously, when they nudge the cabinet. The problem is that most of us have bad intuition about just how hard a nudge it should take to affect the ball. In VP, we're all used to a light tap on the space bar making the ball jump about a foot, so we get the idea that the same should hold for accelerometer nudges. If you try this on a real pinball, trapping the ball on the flipper and giving the machine a few pushes, you'll find that a good hard shove won't even budge the ball. The ball will not jump a foot no matter what you do. Pinball machines are heavy, and the balls alone weigh about a pound apiece. It takes a lot of energy to get them to go anywhere. If you try this experiment with a real machine some time, you'll find that real machines don't react with nearly as much zeal as the VP space bar makes you expect.

The right way to solve problems with over-active nudge feedback is, in nearly all cases, simply to turn down the gain. Turn it down until the devices stop interfering. The nudge will feel a little dead at first, especially if you're more calibrated for PC pinball with space-bar nudging than you are for real pinball machines. But it *should* feel that way if you value realism at all. You should still

be able to see an effect, but it should be subtle, and it should take some real cabinet motion to appreciably affect the ball's trajectory.

See "What about interference from the shaker or subwoofer?" in Chapter 36, Nudge & Tilt.

- The second, much less common reason is that the KL25Z isn't secured tightly enough to the cabinet. The device should be attached in such a way that it moves exactly as the cabinet moves, because the whole point is to read the cabinet's motion and pass it to the software as accurately as possible. It should be secured tightly to a rigid surface like the floor of the cabinet. Make sure that the KL25Z isn't loose, and that it's attached to something that can't move around on its own. It shouldn't be attached to a flexible or springy surface.

Some people on the forums have suggested cushioning the accelerometer with foam padding or something like that to reduce the vibration it receives. I don't like that approach, because it actually defeats the purpose of the accelerometer. You **want** the accelerometer to pick up the cabinet's motion - that's what it's there for. If you mechanically isolate it from the cabinet's motion, you'll reduce the accuracy. Sure, if you put it on a bookshelf across the room, it'll stop reacting to the subwoofer - but it'll also stop reacting to the cabinet's motion. What you want is a rigid connection to the cabinet that makes the accelerometer move in lock-step with the cabinet.

Some additional tips:

- Make sure that you haven't ever used Windows joystick calibration on your nudge device. Windows joystick calibration is for joysticks. Nudge devices aren't actually joysticks - they only pretend to be, to make the software setup easier. The Windows joystick calibration wreaks havoc on accelerometers; it makes their readings erratic and non-linear. If you've **ever** run the Windows joystick calibration procedure, or you're not absolutely sure you haven't:
 - Press Windows+R, type **joy.cpl**, press Enter
 - Find **Pinscape Controller** (or your nudge device, if something else) in the list and double-click it
 - Click the **Settings** tab
 - Click **Reset to defaults**
- **Don't** use dead zones in any of the configuration dialogs for nudging or joystick setup. Dead zones can seem at first glance like a way to solve nervous jitter from small vibrations, but if you think about it a little more, it's easy to see what's wrong with the dead-zone approach. Dead zones are inherently non-linear: with a dead zone, you get no response at all up to a threshold, and suddenly you get a big response. What you really want is for a small nudge to produce a small response and a big nudge to produce a big response. Everything should be proportional. If small vibrations are producing too much of a response, what that really means is that the big nudges are also producing too much of a response - in other words, *everything* is producing too big a response, because the gain factor in VP is amplifying everything too much. That's most visible with the small nudges because it's more obvious when those are hyper-exaggerated. The solution isn't to ignore the small nudges, but rather to use them to calibrate the gain factor, by turning down the gain until the small vibrations don't cause noticeable or excessive responses. If you've ever set a dead-zone in the Visual Pinball dialogs or anywhere else, I'd immediately set them all back to zero. The only VP setting you should adjust to get the nudging force right is the **gain**.

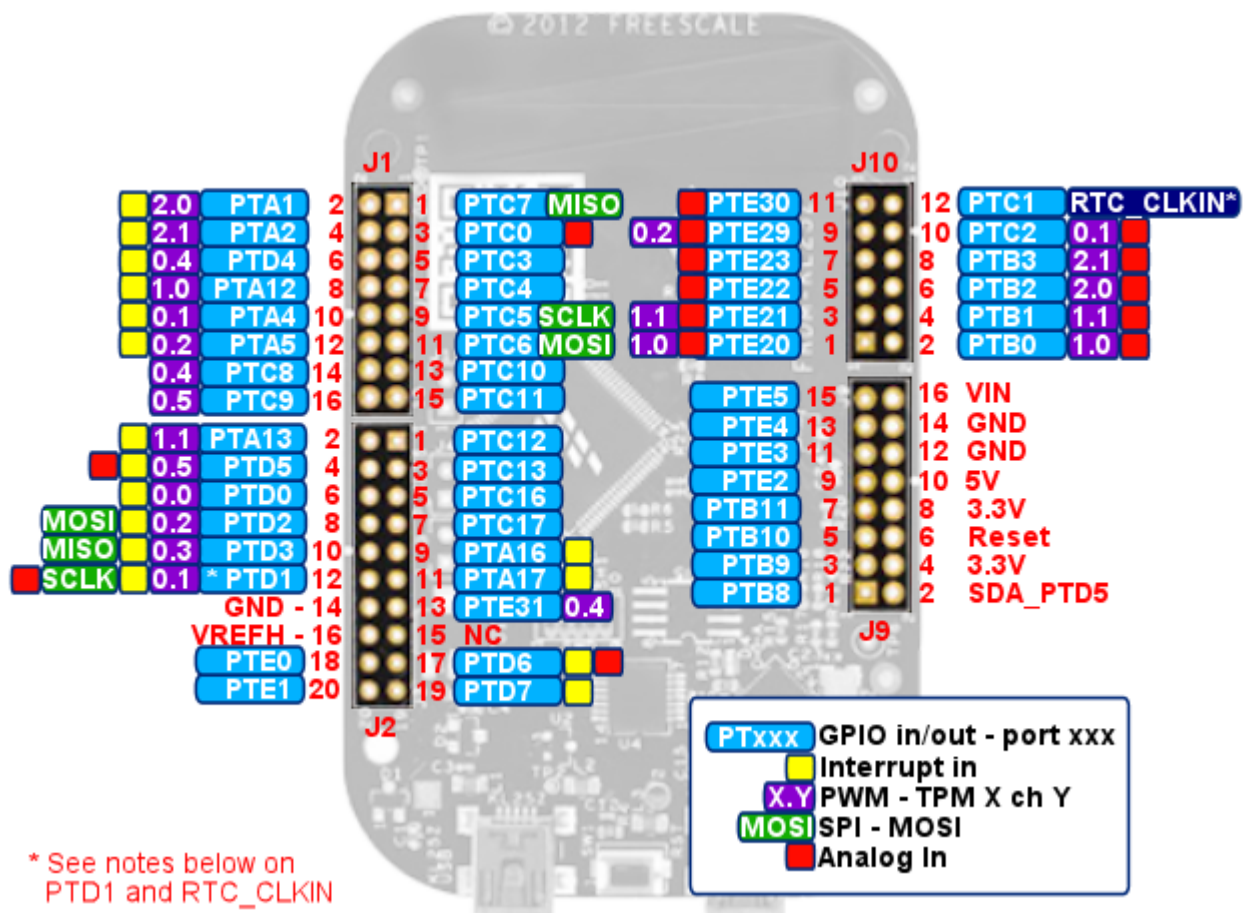
DOF Troubleshooting

See Chapter 46, Troubleshooting your DOF setup.

Appendices



Appendix 1. KL25Z Pin Out

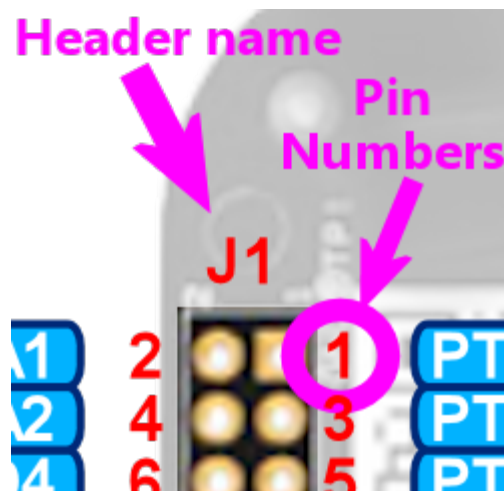


How to read the diagram

The diagram above shows all of the KL25Z's pin connections, labeled by their "header" name and "pin number".

There are four headers - rectangular blocks of pins arranged in a group. They're labeled **J1**, **J2**, **J9**, and **J10**. In the diagram, each header's name is printed just above or below header itself.

The pins on each header are numbered from 1 to however many pins are on the header. The diagram shows each pin's number alongside the pin.



Throughout the Build Guide, we refer to individual pins using notation like this:

J9-4

To find that pin, you go to header **J9** and find **pin 4** on that header.

The reason we use this notation is that the pin numbers by themselves are ambiguous, since every header has a "pin 1" and a "pin 2" and so on. You always have to know which header we're talking about to know exactly which pin we're talking about. So we'll always tell you something like "pin 7 on header J9", or the more compact shorthand "J9-7".

Most (but not all) of the pins *also* have another name of the form **PTA1**. The "PT" stands for "Port", and the "A1" is another arbitrary label like the pin numbers. These range from PTA0 to PTE31. You can see these names in the little blue boxes alongside the pins on the diagram above. These are the names of the "GPIO" or "General Purpose Input/Output" ports, which are electrical connections to the KL25Z CPU itself. These are very important to us, because they're the KL25Z's gateways to the outside world. They're where we can connect buttons, sensors, and feedback devices.

The other markings on the diagram alongside the GPIO port names indicate the "capabilities" of each port. The KL25Z's physical wiring gives special capabilities to certain ports. The capabilities are important to understand if you're writing software for the device or working it into a hardware design, but you won't have to worry about these if you're using the Pinscape software, since it already takes all of this into account. The Pinscape Config Tool will guide you to the proper port selections whenever special capabilities come into play.

PTD1 and the blue on-board LED

The port labeled PTD1 needs some additional explanation. It's a GPIO port, but it's also hard-wired on the KL25Z to the blue segment of the KL25Z's on-board status LED - the RGB LED that normally shows the Pinscape firmware's current status by flashing various color patterns (Chapter 90, KL25Z Status Lights).

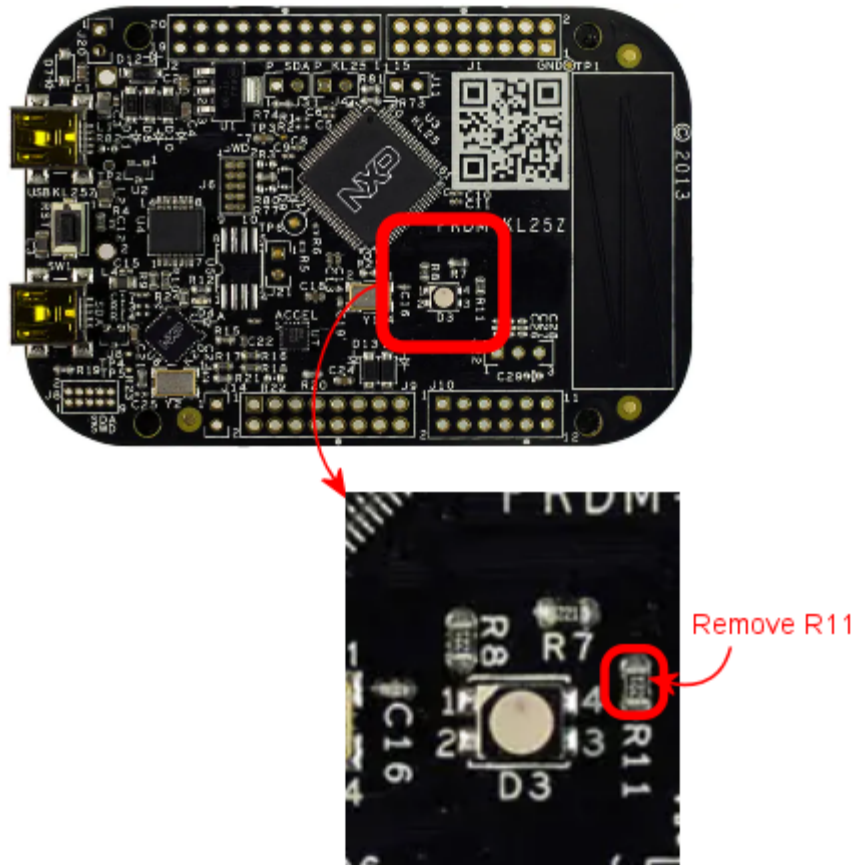
You can use PTD1 as a feedback device output port if you wish, simply by assigning it to an output port row in the Pinscape Config Tool. If you do, its output port use automatically supersedes the status light usage in the Pinscape firmware, so there won't be any conflict at the software level. However, be aware that the on-board wiring to the blue LED will still be there. The blue LED wiring is a fixed, physical feature of the KL25Z board that can't be changed in software. This means that the blue LED segment will turn ON whenever the PTD1 pin is at a "low" voltage, and OFF whenever PTD1 is "high". So the blue LED will light up when the feedback port is OFF and turn off when the feedback port is ON. You can just ignore that if it doesn't bother you, or you can modify the KL25Z as described below to sever the on-board wiring to the LED.

You can also use PTD1 as a button input port, again, simply by assigning it in the Config Tool. As with using PTD1 as an output port, using it as a button input port will automatically supersede its use to control the Pinscape status LED indicator, so the blue segment of the status light will stop functioning. And as with using PTD1 as an output port, using it as an input port won't actually remove the physical connection between the PTD1 pin and the blue LED. The effect as an input port will be that the blue LED will turn on whenever you press the connected button. That should be harmless, but if it bothers you, you can use the procedure below to sever the internal wiring to the blue LED.

I wouldn't attempt to use PTD1 for any other purpose (such as a plunger input, TLC5940 output, etc) *unless* you sever the LED wiring as described below. The LED wiring shouldn't interfere with using PTD1 as a feedback output port or button input port, but it could create problems for other uses, since it places an electrical load on the pin in addition to whatever else you're connecting. That could definitely interfere with using the port as an analog input or for any high-speed signal I/O.

Severing the on-board wiring between PTD1 and the blue LED: To break the connection to the on-board LED and turn PTD1 into a true general-purpose port, remove resistor R11 on the KL25Z board. R11 is a small surface-mount resistor on the top of the board; see the diagram below to help find it. It's labeled with the text

"R11" next to the part, so double-check the label to make sure you've identified the right part.



R11 isn't too hard to remove, since it has a fair amount of empty space around it on the board. My approach to removing it would be to use a pair of forceps to take hold of the part, and then touch a soldering iron to the part to melt the solder, pulling away gently with the forceps until the part pulls free. You could also just use the tip of the soldering iron to apply some gentle sideways pressure to the part while heating it. Continue pressing it until it comes loose, then flick it away so that it doesn't reattach as the solder cools.

If you do sever the blue LED wiring, the blue component of the Pinscape status indicator light will obviously stop working. That will disable all of blue elements of the flash patterns, and will change purple elements to red.

RTC_CLKIN (PTC1, J10 pin 12)

Port PTC1 has a special assignable usage, as the reference clock signal input for the real-time counter, which the KL25Z documentation refers to as RTC_CLKIN. This provides an external reference clock signal (a square wave running at a specific frequency) for the KL25Z's on-board real-time module. The RTC is designed to perform time-keeping functions, particularly related to scheduled events.

The Pinscape software doesn't currently use the RTC module, so this pin is free for use as an ordinary GPIO pin.

If any features are ever added to the Pinscape software that require the RTC module to be activated, this pin will have to be used for the clock input whenever those features are activated. I don't contemplate any such features currently, and even if some future use does become interesting, it would be optional - you'd only have to reassign PTC1 to that usage if you enabled the new feature.

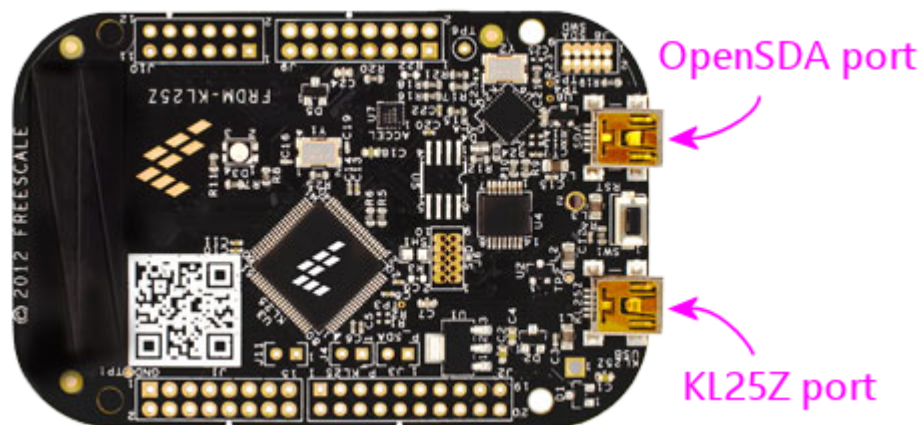
(The pin is actually hard-wired on the KL25Z to a pin on the separate SDA CPU - the one that runs the boot loader, for installing new firmware versions. This is designed so that the SDA CPU can be programmed to generate the required external clock signal for the RTC module. The mbed version of the boot loader unconditionally generates that signal on this hard-wired connection, so if the mbed boot loader is installed, PTC1 can't be used as a GPIO pin - the clock generator feeds a 32768 Hz square wave into this pin, making it effectively impossible to use for anything else. Fortunately, the standard PCMicro boot loader that comes with the KL25Z doesn't do this; it leaves the connected SDA CPU output port in a high-Z state, which effectively turns the hard-wired connection into an open circuit and thus allows us to use the GPIO pin normally.)

SDA_PTD5 (J8 pin 2)

This is another unusual pin that can't be used for any Pinscape function.

Even though the name makes it sound like a GPIO pin, this pin doesn't connect into the KL25Z CPU at all. Instead, it connects to a separate chip on the board where the boot loader resides. "SDA" is Freescale's name for their boot loader system, which is why this pin is labeled SDA_PTD5 instead of just plain PTD5. Confusingly, the boot loader chip has its own set of GPIO ports, with basically the same naming convention as the KL25Z's ports. That's why the name looks like a GPIO pin - it *is* a GPIO pin, just not a KL25Z GPIO pin. That means that it's not usable for Pinscape purposes, since the Pinscape software has no way to access the pin.

USB Ports



(Note that the ports are labeled in fine print on the **bottom** side of the KL25Z, so you don't have to just remember which is which.)

The "KL25Z port" is the one that's connected to the microcontroller proper. This is the port that the Pinscape software uses for its USB joystick, keyboard, and LedWiz emulation interfaces.

The "OpenSDA" port *isn't* connected to the microcontroller. It's connected to a separate processor on the board that's there for the sole purpose of downloading firmware into the microcontroller's flash memory. That separate processor isn't involved in any of the Pinscape functions, so the OpenSDA port doesn't need to be connected at any time except when installing new firmware. But it's also harmless to leave it plugged in all the time - leaving it plugged in doesn't interfere with normal Pinscape operations. The OpenSDA processor just sits there emulating a thumb drive awaiting firmware downloads.

About the KL25Z

The KL25Z is a single-board computer, incorporating a 32-bit ARM-architecture CPU (Cortex M0+), 16K SRAM, numerous on-board peripheral devices within the CPU (three PWM modules, a 4-channel DMA controller, real-time clock, SPI and I2C bus interfaces, a USB 2.0 host/device controller, three UARTs, an analog-to-digital converter, a digital-to-analog converter, a flash memory controller, and more), and additional peripherals on the board but external to the CPU (a three-axis accelerometer, a touch sensor, a clock generator, and an RGB LED).

The board also includes an integrated "programmer" module with its own separate USB connection. This is what allows us to download new firmware, such as the Pinscape Controller software, directly from a Windows machine.

GPIO ports and special functions

All told, the KL25Z CPU chip (not the board, but the chip itself) exposes about 80 external connections. About 50 of these connections are exposed through header pins on the KL25Z board. The rest are mostly dedicated to connections with the on-board peripherals; these aren't exposed externally as they can't be reassigned. That's part of why you find the gaps in port numbering on the exposed pins.

Most of the external header pins on the KL25Z are wired to GPIO ports ("general-purpose I/O"), which can be connected to external electronics to send digital signals to and from the KL25Z CPU. The diagram above shows the layout of the header pins and the GPIO port assignments. The pin layout can't be changed by the software; the pins are physically wired this way on the board.

In addition to the basic GPIO functions, many of the pins can also be connected, under software control, to one or more of the on-board peripherals. As with the GPIO port assignments, the set of these possible connections for each pin is hardwired: each pin can only perform certain special functions. The diagram shows which special functions are available on each pin. The software determines which special function a given pin performs at any given time using another on-board peripheral called the "multiplexer". This is essentially a switchboard that connects pins to special modules. At any given time, a pin can only be connected to one module. For example, a pin that's PWM-capable only generates PWM signals when the software tells the multiplexer to connect the pin to the PWM module.

PWM limitations

The KL25Z hardware has some rather complex limitations on its PWM outputs. The diagram tries to make these evident, but you have to understand the nature of the limitations to make sense of the information in the map.

The KL25Z generates PWM signals (pulse-width modulation) through on-board peripherals called TPMs (Timer/PWM Modules: yes, it's a nested acronym). The CPU has three of these modules: TPM0, which has 6 channels; TPM1, with two channels; and TPM2, with another two channels. That's ten channels total across the three modules.

If you look at the diagram, you'll see that there are 23 GPIO pins marked as PWM-capable. This brings us to the first limitation. You can't actually have 23 separate PWM outputs, despite the 23 PWM-capable pins. The PWM signal reaching the pins has to be generated in the TPM module channels, and as we've seen, there are only 10 of these channels in all. Take a look at the map and find PTA1 and PTB2. You'll see these are both marked "2.0" in their PWM boxes. That means both of these pins are

assigned to TPM2 channel 0. You *can* assign both of these channels as PWM outputs, but if you do, you won't be able to control them independently, because they both get their signals from TPM2.0. Change the brightness on one pin and you'll change it to the same setting on the other pin. So despite the 23 PWM-capable pins, you can really only use 10 of them as PWM outputs at once.

The subtler limitation is in the pulse frequency generation. The **duty cycle** of a PWM output is controlled by its **channel**. However, the **period** of a PWM output is controlled by its **module**. All of the channels on a given module share the same period (or, equivalently, frequency). Many PWM applications don't care very much about the frequency, but the frequency is critical for some purposes. For example, if we're using the PWM signal as a clock generator for an external device, we need an exact frequency setting. This means that if we need two such clocks at different frequencies, we have to assign the two functions to pins that map to separate TPMs, to given them independent control over the frequency.

The ability to set independent frequencies isn't just academic. It actually matters for Pinscape if you're using the expansion boards and/or the IR functions:

- The expansion boards use a PWM output to generate the master clock signal for the TLC5940 chips (called the "grayscale clock" in the TLC5940 data sheet). This master clock frequency isn't set in stone, but it's constrained by a number of factors: it has to be fast enough that the strobing effect isn't visible to the eye, and its upper limit is constrained by the KL25Z's data transmission speed, the ability of the physical wiring to conduct signals reliably, and the TLC5940 chip's design limits. For the full technical details, refer the TLC5940 driver source code in the Pinscape firmware (TLC5940/TLC5940.h in the source tree). The setting as of this writing is 350 kHz.
- The IR transmitter uses a PWM channel to generate the IR carrier frequency. This frequency is determined by the IR protocol being used for a given transmission, so it can vary from one transmission to the next. The protocols currently implemented use carrier frequencies ranging from 36kHz to 40kHz.

The requirements for these features to use specific PWM frequencies, combined with the limitations of the KL25Z's TPM hardware, has two important practical implications:

- First, the TLC5940 interface's master clock port and the IR transmitter's data output port **must** be assigned to GPIO ports that use **separate** TPM units. Each function requires the ability to set its own frequency independently of other PWM usages, and since the frequency of a PWM output is set at the **TPM unit** level, each function must have free control over an entire TPM unit. Therefore each must be assigned to a separate TPM unit.

The Pinscape Config Tool knows about this requirement, and it will automatically guide you to select appropriate GPIO pins if you're using both functions. It'll warn you if you attempt to assign the two functions to the same TPM unit. You must not ignore these warnings; the device won't function properly if you do.

- Second, if you're using a TLC5940 chip and/or the IR transmitter, **and** you're also using GPIO ports as regular feedback output ports, be aware that the GPIO ports you're using might have their PWM frequencies fixed by the TLC5940 or IR transmitter functions. This might be an issue if you're using booster circuitry with the outputs, because the TLC5940 clock setting is a high frequency that might be too fast for booster circuits. Booster circuits that use optocouplers in particular might not work properly, because most optocouplers have a speed limit of about 80kHz, well below the TLC5940's 350kHz setting.

You can mitigate this by assigning your PWM feedback outputs to GPIO ports that use other TPM units. Refer to the matrix below for the associations between GPIO pins and TPM units.

Here's a matrix showing the available GPIO ports for each TPM unit. Remember, the frequency of a PWM port is set at the **unit** level, so all of the ports associated with TPM Unit 0 will operate at the same frequency, all of the ports assigned with Unit 1 will operate a second common frequency, and all of the ports on Unit 2 operate on a third common frequency. Each **channel** represents a separately controllable **duty cycle** that operates at the common frequency for its unit. For example, PTA4 and PTC2 are both on Unit 0, Channel 1, so they represent the same PWM output; they can't be separately controlled, as they share the same frequency (set on Unit 0) and the same duty cycle (set on Unit 0 Channel 1).

TPM Unit	Channel	GPIO ports
0	0	PTD0
	1	PTA4, PTC2, PTD1 (blue on-board LED)
	2	PTA5, PTC3, PTD2, PTE29
	3	PTC4, PTD3, PTE30
	4	PTC8, PTD4, PTE31
	5	PTC9, PTD5
1	0	PTA12, PTB0, PTE20
	1	PTA13, PTB1, PTE21
2	0	PTA1, PTB2, PTB18 (red on-board LED), PTE22
	1	PTA2, PTB3, PTB19 (green on-board LED), PTE23

Why is the pin layout so random?

Blame the Arduino.

The layout is as it is because the KL25Z's designers took Arduino compatibility as the starting point. The Arduino is a popular hobbyist platform that started off on some rather primitive 8-bit hardware. One of the features was a very rigid assignment of particular functions to particular pins - PWM output, analog in, UART RX/TX, etc. The KL25Z is much more flexible in the way it maps these special functions to pins, but it's not infinitely flexible; it too can only expose certain functions on certain pins. In order to achieve the desired pin compatibility with the Arduino, the KL25Z's designers had to arrange the pins in the peculiar order we find them in here. This arrangement allows the pin at each physical position to be assigned to its traditional Arduino role, which allows some Arduino add-on boards to be used directly with a KL25Z. (But be careful if you plan to do this, because there is one other major difference between the Arduino and KL25Z that can seriously affect compatibility: voltage. Most Arduinos run on 5V, whereas the KL25Z is a 3.3V device. The KL25Z will be damaged if exposed to 5V levels on any of its GPIO pins. Many Arduino add-ons generate 5V signals, which makes them incompatible with a KL25Z even if the pin layout matches.)

Appendix 2. VP Customizations Summary

For your convenience in setting up a new cab, restoring a backup, or restoring settings up after a software update, here we present a quick summary of all of the VP customizations recommended throughout the book. You can find more details in the referenced chapters.

What/where are "VP core scripts"?

Where we mention "VP core scripts", we're talking about the shared Visual Basic script files that most VP tables use. These are plain text files with names ending in **.vbs**, and you can find them in the **Scripts** sub-folder of your Visual Pinball install folder. You can use Notepad or a programmer's text editor of your choice to edit them.

Tilt bob

Synopsis: Configure VP to work with a real tilt bob

Details: Chapter 36, Nudge & Tilt

Key encoder device setup: Program your key encoder to use the **T** key on the keyboard for your tilt bob input

VP script setup:

- If you're using VP version 10.4 or later, find the file **NudgePlugIn_mjrAccelAndTilt.vbs** in your VP Scripts folder, and rename it to **NudgePlugIn.vbs**
- If you're using an older version of VP (before 10.4), download my NudgePlugIn.vbs script and put it in your VP Tables folder. (Don't use this version with VP 10.4 or later - use the updated version that comes with VP instead.)

Coin door switch

Synopsis: Configure VP to work with a real coin door switch

Full details: Chapter 40, Coin Door

VP keyboard preferences: Run VP and bring up the keyboard dialog (select Preferences > Keys on the menu). Make sure that **Add Credit** is assigned to the **3** key, and that **Add Credit 2** is assigned to **4**.

Key encoder setup: In your key encoder, assign the coin door switch to the **End** key on the keyboard, if possible, because that's VP's immutable default. If your key encoder can only send joystick button presses, that also works, but you must set the joystick button mapping in VP: bring up the **Keyboard** preferences dialog, find the **Coin door (End)** item, and select the appropriate joystick button number in the drop list.

VP scripts: If you wired your coin door switch the recommended "new" way, as a simple On/Off switch rather than as some kind of "toggle mode" contraption, edit the core VP core script **VPMKeys.vbs** as follows.

Find these original lines:

```
toggleKeyCoinDoor = True      ' If true then...
inverseKeyCoinDoor = False    ' If false then...
```

...and change them to read:

```
toggleKeyCoinDoor = False    ' If true then...
inverseKeyCoinDoor = True    ' If false then...
```


Appendix 3. Plunger Sensor Technical Notes

Tracking a plunger for pinball simulation is a surprisingly difficult technical challenge. At first glance, it looks like a pretty easy problem with some simple solutions: it's just a linear position sensor, and there are some standard approaches for that. But there are two details to pinball simulation that make the problem harder: you need fine spatial resolution, and you need to be able to take readings quickly.

Spatial resolution and sampling speed

How well a plunger performs in a pinball simulator depends upon its spatial resolution and sampling speed. I consider a sensor suitable if it can determine the true plunger position to within $1/100"$, and produce at least 200 such position readings per second.

Spatial resolution refers to how precisely you can locate the current position of the physical plunger. The pinball simulator uses the position reading from the sensor to draw the on-screen plunger on the video display in real time, so that the player can see the on-screen plunger animated in sync with the physical plunger. For that animation to be realistic and smooth, the on-screen position has to track the physical position as finely as the video display can draw it. On an HD screen at playfield size, the spacing between pixels is about $1/50$ of an inch, and on a 4K monitor it's about $1/100$ of an inch. On a full-sized pin cab, the on-screen plunger should be roughly the same size as the real plunger, so if you want smooth, pixel-accurate animation, the sensor has to be able to pick up movement of the physical plunger on the same scale as the size of the on-screen pixels. That requires $1/100"$ precision from the sensor.

Spatial resolution also affects the quality of the physics simulation. All of the pinball simulators read the plunger position via the Windows USB joystick interface, which is capable of representing high precisions. Visual Pinball can handle 16-bit resolution, which translates to one part in 65,536, or about 50 microns (millionths of an inch) over the 3" travel of a physical plunger. That's a lot finer than the $1/100"$ resolution required for smooth animation, and much finer than any virtual pinball plunger sensors currently available (for Pinscape or any other system). The point is that there are practical benefits to working on better sensor designs than we have today, since VP's physics model can take advantage of a lot finer spatial resolution than today's sensors provide.

Sensor speed is also critically important. The plunger moves really fast when you pull it back and release it, so in order to track that motion in real-time, the sensor has to be able to take readings quickly. My measurements with the AEDR-8300 sensor indicate that the plunger in my test rig, which is a standard Williams/Bally plunger with a medium-tension spring, moves at a peak speed of about 4.5 meters per second. That means that the plunger covers its full 80mm travel range in about 25ms. To keep up with that fast motion, the sensor has to be able to take readings at least about every 5ms, or 200 times per second. Sensors with slower sampling rates than that will suffer from "aliasing", where they mistake retrograde motion for forward motion (or vice versa) when the plunger bounces back after hitting the barrel spring. When aliasing occurs, VP can't calculate the speed of the plunger correctly, which screws up the physics simulation. Aliasing can lead to wildly erroneous speed calculations. The Pinscape firmware has special handling to detect and correct for aliasing in slower sensors, but that necessarily loses information about the true instantaneous speed and position, so it's better if the sensor can produce readings frequently enough that the aliasing filter is never triggered.

Potentiometer

This is one of the simplest ways to measure a linear position, and it actually compares pretty well to more complicated approaches. It's used in a lot of industrial applications where you need fast readings without extremely high resolution.

Physically, a potentiometer consists of a fine electrode that moves across a resistive conductor such as graphite - something that conducts electricity, but with relatively high resistance. Resistance in a material like graphite is a function of the distance between measuring points, so as you move the electrode across the graphite, the resistance between the electrode and any fixed point on the graphite varies with the distance between the two points. If you measure the resistance at point A as higher than the resistance at point B, you know that point A is further away from the fixed end than point B.

For our purposes, we use "linear taper" potentiometers, which are built so that the electrical resistance varies linearly with the electrode position. Voltage and resistance are also linearly related (through Ohm's Law), so if we supply a fixed voltage across the fixed ends of the pot, the voltage we read at the electrode will vary linearly with the resistance, which varies linearly with the position of the electrode. The KL25Z has an on-board analog-to-digital converter (ADC) that can sample an analog voltage level and report it to the software as a digital reading.

How accurate is a potentiometer? The one that we recommend in the parts list is readable to about 1% accuracy with the KL25Z ADC. That translates to about 3/100". So it's not quite as good as we need for perfectly smooth animation, and it means that the "jitter filter" in the Pinscape firmware is usually needed to get steady readings when the plunger is standing still.

This could be improved by using a different potentiometers with higher accuracy, but I haven't found anything available that looks like it would do better.

The potentiometer's spatial resolution might not be perfect, but its sampling speed is extremely fast. The potentiometer itself can be sampled in arbitrarily short intervals, as it provides an analog voltage level that adjusts to changes in position effectively instantaneously. Everything at the analog level moves at the speed of electric field propagation in the conductors, which is close to the speed of light. So the limiting factor isn't the analog signal, but rather the ADC sampling time. The ADC works by charging a small capacitor internally, and this takes a more macroscopic amount of time. The fastest possible ADC sampling time on the KL25Z is about 10 μ s, although there's a significant trade-off between speed and accuracy, so we use a slower mode that takes more like 30 μ s for each reading. That's still about 100 times faster than we need it to be (as discussed earlier, we only need readings about every 5ms), so we can take many readings over the sampling period and average them together, to further reduce noise and improve accuracy. Since the January 2020 version, the firmware takes readings continuously, and computes a rolling average each time a result needs to be sent to the PC. Each reading sent to the PC represents the average of 128 ADC samples. This smooths out the noise and makes for highly stable readings.

Quadrature sensors

Quadrature is an electrical engineering term describing two signals that are 90° out of phase with each other. This has numerous applications in signal processing, but the one we're concerned with here is its use for motion sensing. A quadrature motion encoder works by observing transitions between "on" and "off" zones marked along a "measuring stick", using two sensors that are offset slightly from one another. The

measuring stick is more formally called a "scale".

The 90° phase difference in this case refers to the size of the offset between the two sensors: it's arranged so that it's half the size of the marked on/off zones on the scale. If you observe the on/off signal from each sensor as they move across the scale, you'll see the leading sensor make its on/off transitions 90° of phase ahead of the trailing sensor. Which sensor is the "leading" sensor and which is the "trailing" one depends on which direction they're moving across the scale, so you can tell which direction you're moving by observing the phase between the two signals and seeing which sensor is making its transitions first at each step. Or, equivalently, you can look to see if the signals are 90° or 270° out of phase with each other.

The scale can be implemented in many ways, but the common element is that it has to have a series of on/off zones of about equal size across the range of travel. These can be electrically conductive and non-conductive regions, for example, or magnetic north/south poles, or optical black/white or transparent/opaque bars. The specific quadrature sensors that we use in the Pinscape projects are optical sensors, so our scales use the transparent/opaque approach. In other words, our scale is simply a series of black bars printed on transparent film.

A quadrature sensor isn't actually a *position* sensor - it's a *motion* sensor. This type of sensor can only detect when it's moving from one bar to the next on the scale, and the direction of that motion (using the signal phase). If we combine these two capabilities - detecting a change from one bar to the next, and knowing which direction we went during that change - we can keep count of how many bars we are away from our starting point. So if we have a known starting point, and we know the length of each bar, we can infer our current position: our position at any given time is the starting position plus or minus the net distance we've moved since then.

For the plunger application in particular, we have an excellent "known starting point", namely the plunger rest position. It's usually safe to assume that the plunger starts at the rest position, so we take that to be the initial position at startup. All subsequent inferred positions are relative to this starting point.

There's an inherent drawback to using a quadrature sensor as a position sensor by way of the net travel distance, which is the possibility for missed transitions. If the sensor ever misses a transition, the count will no longer be in sync with reality, so the inferred position that we figure based on the count will differ from the actual position by the number of missed counts. Sensors in practice do tend to miss some fraction of transitions, so the error between inferred and actual position tends to increase over time. To mitigate this in our application, the Pinscape software can be set, optionally, to reset the inferred position to the starting (plunger rest) position after any extended period without any motion. Just as it's usually safe to assume that the initial position at system startup is the rest position, it's a good bet that a motionless plunger is in equilibrium at its rest position, since it's difficult to hold it perfectly still anywhere else for any length of time, and there's no reason to do so even if you could.

Interfacing a quadrature sensor to the KL25Z is straightforward. We treat the two sensor output channels as digital inputs (high/low signal levels, corresponding to the on/off zones in the scale), and set them up to trigger interrupts. The interrupt handler for each channel uses a lookup table to figure the direction increment represented by the current channel signal levels, adding or subtracting one from a running counter.

A fast-moving plunger can result in extremely rapid signal transitions on the channels - fast enough to overwhelm the KL25Z's interrupt response time. The quadrature sensor software interface in the Pinscape firmware uses custom hardware

interrupt handlers to optimize the response time, and with that it requires $6.5\mu\text{s}$ to process each channel interrupt. When considering new quadrature sensors, we have to calculate the signal rate at peak plunger speeds (about 4.5 mm/ms) to make sure it doesn't exceed the $6.5\mu\text{s}$ response time, or equivalently, about 150kHz . In many cases, the quadrature sensor itself might be the limiting factor - most of the ones I've looked at quote peak signal rates in their data sheet far below 150kHz (it's usually closer to 20kHz to 30kHz).

AEDR-8300

The AEDR-8300 is a reflective optical sensor. It has an LED emitter that shines a narrow light beam towards the scale, and a pair of photoreceptors that capture the light reflected back from the scale. The photoreceptors use the usual quadrature arrangement where they're slightly offset from one another.

The AEDR-8300 product line has parts available with different bar spacings. The particular sensor we use has 75 line-per-inch bars, meaning that each black/white pair is $1/75"$ wide. The nature of quadrature sensing means that we can tell our position to within $1/4$ of a line pair, so we effectively get $1/300"$ resolution.

At peak speeds, the 75 LPI spacing results in quadrature pulses from the sensor about every $19\mu\text{s}$. That's safely above the $6.5\mu\text{s}$ interrupt response time I've measured on the KL25Z. Note that the finer-pitched sensors in the AEDR-8300 line would likely overwhelm the KL25Z, and probably wouldn't be able to keep up with the peak plunger speeds anyway, as these sensors have their own maximum signal rates that are below the KL25Z's limit. But that's okay, since the 75 LPI spacing gives us such high precision that we really have no need for a finer pitch.

Linear image sensors

The first Pinscape plunger sensor was based on a linear image sensor, the TSL1410R (and the similar TSL1412S). These sensors are no longer available. The TCD1103 works on a similar principle, but it requires a focusing lens, which makes it more complex to set up.

The principal of operation of these sensors is pretty simple. The sensor consisted of a single row of light sensor pixels about the same length as the plunger travel distance. We arranged the sensor so that the row of pixels ran along the axis of the plunger travel, and placed it directly adjacent to the plunger, with the pixels facing the plunger. We put a light source on the opposite side of the plunger, facing the pixels. This caused the plunger to cast a shadow on the pixel array. We'd then read the pixel array and find the location of the shadow, by looking for an edge between a light area and a dark area in the image. The edge tells us where the plunger is currently positioned.

In principle, a sensor of this type could achieve spatial resolution equal to its pixel size. In practice, though, the shadow cast by the plunger isn't perfectly sharp. Shadows always have a little fuzzy area when the light source isn't a perfect point source, because the light source is only partially blocked at the leading edge of the shadow. So we can't tell exactly where the plunger was; we have to guess based on the midpoint of the shadow. The best possible resolution in my setup is about $1/50"$. That's a bit below the ideal needed for pixel-level resolution on the TV display, so there's a little bit of jitter (by a pixel or two) as the guess about the shadow position would vary from frame to frame.

The TCD1103 can perform much closer to its theoretical pixel limit thanks to the requirement for focusing optics.

The Pinscape software uses a simple edge-detection algorithm with these sensors, where it looks for a region with a rapid transition from a run of bright pixels to a run of dark pixels. This algorithm inherently compensates to some extent for exposure levels, since it works based on the difference in brightness across pixels rather than looking for absolute brightness levels.

TSL1410R/TSL1412S

The TSL1410R happened to be perfectly designed for this application. Its pixel row was 3" long, with 1280 pixels across the file. That's 400 pixels per inch.

The TSL1412S was almost exactly the same, with the only difference being a slightly longer sensor window and a correspondingly larger number of pixels (1536). The pixel spacing was the same on both sensors, and they were identical in their electronic interface.

These sensors work by using photoreceptors to charge capacitors. The electronic interface lets the host microcontroller connect the individual integrating capacitors to an analog output port, one at a time through a shift register. The capacitor charge is an analog voltage level, so we use the KL25Z's ADC to sample each pixel's charge level and convert it to a digital reading.

The KL25Z has the capability to control the ADC through its DMA (direct memory access) controller, which was key to making the image capture fast enough to work as a plunger sensor. The DMA runs concurrently with the CPU, so the firmware was able to start the DMA reading process and then return to other work while the pixels were clocked into memory by the DMA controller. This allowed a full reading to be taken in about 2.5ms, or 400 frames per second, which is fast enough to keep up with the peak plunger release speed.

TCD1103

This is a tiny CCD sensor made by Toshiba, with 1500 pixels arranged in a single linear file. It works using the same edge detection algorithm as the TSL1410R/TSL1412S, but its pixel window is only about 8mm long, so it requires a lens to focus a reduced image of the plunger on the sensor. If the lens is properly aimed and focused, this sensor can resolve the edge at the end of the plunger to a single pixel, so it achieves spatial resolution of nearly its full 1500 pixels across the 80mm travel range of the plunger, which translates to about 1/400" resolution.

The electronic interface to this sensor is similar to that of the TSL141x sensors. Like those sensors, the TCD1103 has an electronic shutter function that moves the pixel charges onto capacitors, which the microcontroller reads out by sampling an analog voltage output one pixel at a time. With DMA transfer, the KL25Z can transfer the pixel file in about 3ms.

Distance and proximity sensors

The original commercial plunger kits (e.g., the Nanotech Mot-ION controller, and the various generations of the VirtuaPin plunger kits) and all used IR-based proximity sensors. IR proximity sensors work by emitting an IR light signal and sensing the brightness of the reflection from a nearby object via a photoreceptor.

Technically, proximity sensors are only meant to detect *proximity* - a yes/no question, "is an object within range of the sensor?" However, most of these sensors don't answer the yes/no question directly, but rather just give you their raw analog brightness reading, leaving it up to the application to interpret that by comparing it to a calibrated threshold. The commercial plunger kits don't just treat

the brightness as a binary value above or below a threshold, however. They instead interpret it as a continuous quantity that correlates to the distance to the target (which, in this case, is the end of the plunger rod). At a basic physics level, the brightness of a light source (or of a reflection) varies inversely with the distance to the source, so by applying some math to the brightness reading, we can infer the distance.

Proximity sensors as a class aren't meant to be used this way, though. Their brightness sensors aren't designed to be fine measurement instruments, and attempting to interpret the brightness reading as a proxy for distance doesn't always yield very good results. The relationship between reflected brightness and distance is inherently non-linear at the physics level, even with an idealized theoretical model. It's worse in a practical setup, since the sensor is subjected to interference from stray light, reflections from other surfaces besides the target, variations due to ambient temperature, and electronic noise. The early commercial plungers, which were based on Sharp analog IR proximity sensors, could only achieve a spatial resolution about 1/2".

A newer IR proximity sensor chip, Vishay's VCNL4010, performs quite a lot better. This chip has its own on-board ADC for the brightness signal (the older Sharp chips produced an analog voltage that had to be sampled externally, on the microcontroller), which seems to make for much higher resolutions. This is reportedly the sensor used in the VirtuaPin v3 plunger kit, so I recently (May 2021) added support for it to the Pinscape firmware, to help out some VirtuaPin customers who had requested it because they switched to the Pinscape firmware on their controller boards. This sensor is actually pretty good: in my testing, it reliably resolved distances to less than 1mm over most of its range. (It gets coarser at the "far" end where the plunger is farthest away from the sensor; this region is the most challenging for the sensor because very little light is reflected back at that distance.) That's not as good as some of the other sensors (potentiometer, quadrature, linear imaging), but it's good enough to be usable. The tricky part of this sensor is that its response curve (the relationship between measured brightness and object distance) is a little bit erratic. But it's close enough to a power law relationship that the distance conversions come out looking plausibly linear, if you don't look too closely.

There's a related but different type of sensor that uses reflected IR light to actually measure distance - by design, not just incidentally. These are called "time-of-flight" sensors, because they measure distance by timing the round-trip time for light pulses to be returned from the target, instead of just measuring the brightness of the reflection. In principle, this should yield much more precise distance measurements, because the time measurement isn't affected by the reflectivity of the target, which is obviously a huge factor for the brightness sensors. In addition, the underlying physical quantity being measured - round-trip travel time of the IR photons - has an inherently linear relationship with the distance, so the conversion from measured quantity to distance is straightforward and doesn't lose precision or magnify uncertainties, the way that the inverse power-law relationship does for brightness-to-distance conversions.

The Pinscape software supports the current best-of-breed sensor in the time-of-flight class (as far as I've found), the VL6180X. Unfortunately, in my testing, this sensor is inferior to the VCNL4010, despite the more sophisticated technology. The VL6180X has a nominal resolution of 1mm (1/25"), which I'd consider just barely usable. Unfortunately, that's only the nominal resolution; the actual accuracy in my measurements is more like 5mm to 10mm. That's too coarse to be usable, in my view; it gives you some semblance of plunger operation, but isn't good enough for smooth animation, and definitely isn't good enough for skill shots. The sensor is also too slow; it takes it at least 14ms to produce a sample, in its fastest and least

accurate averaging mode.

I'm planning to keep an eye out for new time-of-flight sensors coming onto the market, since this seems like a promising design approach that could eventually yield high-precision non-contact distance sensors. But right now there doesn't seem to be a production chip available that performs well enough for our purposes. By the same token, given that the VCNL4010 is already passably good, it would be worth monitoring developments in that product line; it wouldn't take a huge amount of improvement for that type of sensor to become a top choice.

Bar code absolute position sensing

This is an experimental approach that I've been looking at but haven't yet made work well enough. The idea is to do **absolute** position sensing using an optical Gray code and a suitable small image sensor. We create an optical scale with a Gray code printed on it with a unique code in each position across the 3" plunger travel range, then we use an optical sensor to reach the code at the current position. We translate the bar code by looking it up in a table to get the absolute position.

The principle here is simple, and a suitable sensor is available: TSL1401CL. As you might guess from the name, this sensor comes from the same series as the late, great TSL1410R that we mentioned earlier, but it has a much smaller window - only about 128 pixels over 1cm. That makes it unsuitable for the straightforward "shadow edge" method we used with the larger sensor. But 128 pixels is more than enough to read a bar code. With 128 pixels to work with, we could easily come up with a bar code that could represent perhaps 20 bits. That would be much more than we need; just 10 bits would be enough to encode 1000 unique positions along a scale. If we spread 1000 codes over 3", we'd be able to resolve about 300 positions per inch. That's at the high end of the resolution requirements we laid out earlier.

This all works nicely in principle, and the Pinscape firmware already has support for a form of this, since I've done some experiments with it. The challenge is making the optics work. I haven't been able to find a way to get a sharp enough image of the bar code to read it reliably at the required resolution. So the open area of research is how to arrange a lens or other optics to get a good enough image on the sensor.

If I can get this working, I think it would make a really great sensor option. It should have the excellent spatial resolution and sample-to-sample stability of the quadrature sensors, plus the absolute position sensing of all of the other types, which would be a powerful combination.

Rotary absolute position sensing

Absolute sensing like that described above is available in commercial position sensors that do *rotary* position sensing - that is, they measure the turn angle of a shaft, rather than a linear offset. There are some magnetic rotary absolute sensors available with very high angular resolutions.

The trick is to translate the linear motion of the plunger into a rotational angle.

One possibility is a lever between the plunger and rotating shaft. A lever connects to a shaft at one end and to the plunger at the other end; moving the plunger rotates the shaft through the lever. One complication is that the distance between the shaft and plunger would vary over the plunger's range of motion, so a sliding connector at one end would be required. The other complication is that the rotational angle wouldn't vary linearly with the plunger's position - it would be sinusoidal. The sine curve approaches closer to linearity the longer you make the lever, so one solution

could be to make a very long lever. On the other hand, that would require a very high-res sensor, since you'd only be using a small fraction of its angular range. A better solution might be to use calibration in the software to figure out where you are on the sine curve, and translate mathematically from the sine curve to the linear position. That's easy in principle, but it seems challenging to make the mechanics of the calibration user-friendly.

Another possibility is to translate the linear motion to angular motion through a belt that wraps around a pair of wheels. The belt moves with the plunger, and as it moves, it turns the wheels. The angular sensor is connected to the axle of one of the wheels. This would yield a very straightforward linear relationship between position and angle, so it wouldn't have any of the calibration problems of the lever approach. This seems like a pretty straightforward mechanism, but I haven't tried to realize it physically yet. I think the main challenge would be ensuring a solid connection between the plunger, belt, and wheels, so that the plunger motion directly translates to wheel motion without any play or hysteresis. Any mechanical play in the system would manifest as loss of precision in the readings, so this would all have to be very solidly connected.

If the mechanics could be worked out, selecting a suitable absolute rotary encoder and connecting to the software should be relatively simple. There are a number of high-res magnetic encoders available with 12-bit or higher resolution, which means 4096 counts per turn. With the wheel arrangement describe above and 1" diameter wheels, the 3" of plunger motion would turn into almost one full turn, so we'd get perhaps 75% of the range of the sensor. That means we could achieve as many as 1000 counts on the sensor per inch, or 1/1000" linear resolution. That would be far higher than any of the other sensors. Interfacing one of these sensors to the software should be pretty easy, as the newer ones use modern microcontroller-friendly interfaces like SPI or I2C.

Appendix 4. Tables with MagnaSave Buttons

Most virtual cab builders include a second set of flipper buttons, located just behind (or sometimes below) the regular flipper buttons. Virtual pinball people usually call these the "MagnaSave" buttons, named after the trademarked ball-save feature that Williams used in a series of machines in the 1980s. The extra buttons on those machines activated magnets under the playfield that could catch the ball when it was about to enter one of the side drains.



Examples of extra flipper-type button positioning on real tables. Clockwise from top right: Black Knight (Williams, 1981), Special Force (Bally, 1986), Nip It (Bally, 1972), Jolly Park (Spinball, 1996). The in-line horizontal placement used in Black Knight is the most common, and most people think of it as the standard because Williams used it on all of their later machines, but some people find the extra buttons easier to reach with the vertical or diagonal arrangements. Note that all of the machines pictured here use the older, wider style of side rails that required cut-outs for the flipper buttons. The newer WPC-style side rails are much narrower and don't overlap the flipper button area, so you won't have to cut any holes if you're using modern parts to build your cab.

When I was building my cab, I was torn between, on the one hand, wanting to include the extra buttons for the sake of playability for those tables that need them, and on the other hand, wanting a more authentic look to the machine. The extra buttons aren't at all common on the real machines built in the 1990s, or indeed those built in any era, so I was leery about the aesthetic impact of including them. Some virtual cabs I've seen go so overboard with extra buttons that they look more like video games or slot machines than pinballs, and I didn't want to go too far down that road.

To help decide, I figured I should get a better idea of how many games would actually be affected if I didn't include the buttons. I knew there were a few MagnaSave titles from the 1980s, but I didn't know the exact number, and I didn't know if there were any other machines over the years that had similar buttons for

other features. So I started digging through IPDB and asking around on the forums.

The results surprised me. A little IPDB research reveals that the actual MagnaSave games amount to a rather modest seven titles. That much was in line with my expectations. What I didn't realize was how many other games were out there with other extra-button features besides MagnaSave. It turns out that dozens of real machines over several decades had extra flipper-like buttons. We really should stop calling them MagnaSave buttons, since the actual MagnaSave machines are only a small minority of games with similar buttons. But I have to admit that I don't have any better ideas for a name; "extra side buttons" just isn't as catchy.

The other thing I learned was that a rather large number of Visual Pinball tables rely on the nominal MagnaSave buttons to substitute for other, unusual types of special controls that were used in the original tables they're re-creating. I didn't realize before how many pinball machines have their own unique extra controls. This really shouldn't have surprised me, since pinball designers always want to make their games stand out from the crowd, and novel interactive controls are a good way to do that. But if the whole point of these extra controls is to be unique, they present a problem to the virtual cab builder, which is that we can't possibly include physical replicas of all of the novel controls from hundreds of different machines in a single virtual cab. There's just not room. And so, Visual Pinball table authors have adopted the convention that the MagnaSave buttons should serve as stand-ins when a table has unique controls beyond the standard flipper, Start, and launch buttons. This is another point in favor of including MagnaSave buttons on your virtual cab.

The results of my survey are presented in the list below. In the course of compiling this, it became apparent that there are two separate categories of MagnaSave usage that are important to virtual pin cab people, so the list is divided accordingly:

- **Originals** are tables where the original arcade version featured extra flipper buttons on the side of the cabinet. For these tables, the MagnaSave buttons on your virtual cab will serve as accurate replicas of the extra side buttons on the real arcade game, providing very much the same playing experience.
- **Virtuals** are tables that didn't have MagnaSave-style buttons in the original arcade versions, but that do have some *other* kind of extra controls - extra buttons located somewhere other than alongside the flipper buttons, or some entirely different kind of control, such as a lever or knob. Visual Pinball can't assume that you have anything like those other controls on your pin cab, but it does assume that you have MagnaSave buttons, so Visual Pinball table authors have adopted the custom of using the MagnaSave buttons as substitutes for special controls. That changes the playing experience, but it at least makes it possible to fully play the game.

For my own build, this list settled the question for me, easily in favor of including the extra buttons. As you can see, a lot of games take advantage of them. But there's no reason you have to come to the same conclusion; you might read through the list and conclude that you're just not interested in enough of the titles that use the extra buttons. That's why I thought it was useful to present a list of specific titles, to make it more concrete than just a vague claim that there are lots of such games.

Originals	
Game	Function
Black Knight	MagnaSave

Black Knight 2000	MagnaSave
Blackwater 100	Buttons move flashing selection and lower start gate
BMX	Flex-Save
Bone Busters Inc.	Controls extra flipper on upper ramp
Defender	left is "Reverse" (left lane kickback), right is "Smart Bomb" (scoring feature)
Devil's Dare	Ball save (manually activated outlane kickback)
Dungeons & Dragons	Flex-Save
Fireball II	Raises center post between flippers
Freedy: a Nightmare on Elm Street	Raises center post between flippers
Grand Lizard	MagnaSave
Harley-Davidson (1999)	Raises center post between flippers
Hardbody	Flex-save
Haunted House	Secondary flipper buttons
Heavy Metal Meltdown	Lane change
Johnny Mnemonic	Control magnetic data glove
Jolly Park	Mini-playfield magnetic diverters
Judge Dredd	Mode selection
Jungle Lord	MagnaSave
Medusa	Raises "Shield of the Gods" center post between flippers
Nip It	Activates "balligator" (diverts ball to special scoring lane)
Party Animal	Left button changes P-I-G lights, right changes O-U-T lights; scroll through high scores in attract mode
Pharaoh	MagnaSave
Revenge From Mars	Mode selection, aligns crosshairs in attack mode
Rocky	"Instant Win" (scoring feature)
The Rolling Stones LE (Stern)	Activates pop-up outlane blockers
The Shadow	Control ramp diverters ("phurbas")

Sharkey's Shootout	Raise ball-saving posts in outlanes, and between flippers when pressed simultaneously
The Simpsons Pinball Party	Unknown
Solar Fire	MagnaSave
Special Force	Rocket buttons
Speakeasy	"Sacrifice" scoring feature (cancels cards collected out of sequence at cost of 25000-point score penalty)
Spirit	Outlane ball-save flippers
Star Light	Lane change
Starship Troopers	Controls small secondary flipper
Star Wars Episode I	Mode selection
Strange Science	Manually controlled outlane kickback
Striker	Side-to-side passes (scoring feature)
Viper	Buttons control turret in center of playfield when ball enters it (left button reverses direction of rotation, right fires ball)
Volcano	Left controls manual outlane kickback; right controls shooter guide
World Cup Soccer	MagnaSave

Virtuals		
Table	Function	Original control style
AC/DC (Stern)	"Fire"	Stern-style lockbar button
Apollo 13	Ball launch	Rotating handle in place of plunger
Austin Powers	"Fire"	Medium circular red "fire" button left of center on top of lockbar
Baby Pacman	Joystick up/down on flippers, left/right on MS	Video game style console with joystick
Black Rose	"Fire"	Rectangular button on lockbar
Caveman	Joystick up/down on flippers, left/right on MS	Joystick on top of triple-deep lockbar

Demolition Man	Launch balls & move crane	Joystick type buttons on handles sticking up from sides of cab
The Getaway: High Speed 2	Gear shifters	Up/down shift lever in place of plunger
Granny and the Gators		Video game style console with large circular "paddle" left/right buttons and "fire" pushbuttons
The Hobbit	"Fire"	Stern-style lockbar button
Jurassic Park	Smart Missile	Gun-like launcher with trigger and large circular red "Smart Missile" thumb button
Last Action Hero	Smart Missile	Gun-like launcher with trigger; large square yellow "Smart Missile" button on front of cab just above gun
Mac Attack (Mr. Game 1990)	Unknown, possibly used in video mode	Handles on either side of cabinet with red buttons on top
Mustang LE (Stern)	Fire?	Stern-style lockbar button
Odisea Paris-Dakar (Peyper)	"El Movimento"	Knobs on side of cabinet near flipper buttons
Riverboat Gambler	Place bets on roulette game with flippers+MS	Four rectangular buttons on top of lockbar
Sir Lancelot (Peyper)	"El Movimento"	Knobs on side of cabinet near flipper buttons
Star Trek (Stern 2013)	"Fire & Select / Punch It!"	Stern-style lockbar button
Star Wars (Data East)	Fire, Shift	"Shift" handle (pulls up) in place of plunger, with Fire thumb button on left side
World Cup '90 (Mr. Game 1990)	Unknown, possibly used in video mode	Handles on either side of cabinet with red buttons on top
Wolf Man (Peyper)	"El Movimento"	Knobs on side of cabinet near flipper buttons

Notes on the special controls

Stern-style lockbar button: Many of the Stern titles from the mid "aughts" (around 2005) to present feature an extra button on the top of the lockbar that activates special features at certain points during the game. We usually call it the "Fire!" button because that's how it's labeled on several of the titles. It's so

ubiquitous on newer Stern games that I think it's on the verge of joining the set of standard controls you expect to find on a virtual cab, but so far it's not common. For more, see the notes on the Chapter 34, Fire! button in Chapter 34, Cabinet Buttons.

"El Movimento": A few games from Peyper (a Spanish manufacturer) had a feature known as "El Movimento", which used knobs next to the flipper buttons that moved the playfield. The VP re-creations of these tables approximate this by mapping the MagnaSave buttons to a special non-tilting nudge function.

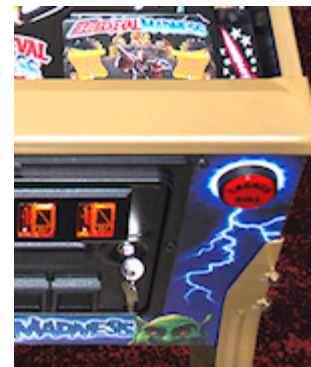
Appendix 5. DOF Config Tool Device Descriptions

When you set up your cabinet with DOF, you'll want to use the DOF config tool to map out the physical devices in your cab. This lets the Visual Pinball table scripts know how your physical devices relate to the events in the game.

The Config Tool lets you make these assignments by telling it the type of device attached to each output controller port. The tool has a pre-defined set of standard types, so you just choose the matching type for each device. These are all in terms of pinball abstractions, because that's how DOF knows to connect them to events in the game. Most of these pinball device types are self-explanatory, but some of them are obscure, and some use virtual pin cab jargon that might not make sense even if you're a long-time pinball fan. To help clear up the confusing parts, below you'll find a list of all of the device names that appear in the Config Tool, along with explanations of what they mean and how they're usually implemented in a virtual cab.

Start Button: The lamp inside the Start button on your cab's front panel. On most games, this either stays lit the entire time, or lights up or flashes when there are coin credits available to start a new game. See Chapter 55, Button Lamps.

Launch Button and **Authentic Launch Ball:** Both of these options are for the lamp inside your Launch Ball button, if you have one. They're both meant to be used with the same button light, but they behave slightly differently. You get to choose the one whose behavior you prefer. See Chapter 55, Button Lamps.



The difference between the two options is pretty subtle:

- The regular "Launch Button" setting turns on the button lamp in most games whenever a ball is in the plunger chute.
- The "Authentic Launch Ball" setting only turns on the button lamp when you're playing a machine that *originally* had a Launch button in place of a plunger. On those machines, it reproduces the original behavior of the Launch button light on the real machine. When you're playing a game that had a plunger in the original arcade version, this setting generally leaves the Launch button light off.

For example, suppose you load *Earthshaker* into Visual Pinball. The real *Earthshaker* has a plunger, not a Launch button, so the "Authentic Launch Ball" light will stay off the whole time you're playing this table. Now suppose you load *Medieval Madness* into VP. Real *MM* machines have a Launch button instead of a plunger. In this case, "Authentic Launch Ball" will control the lamp using the same pattern it would on a real *MM* machine. In contrast, the regular "Launch Button" setting would turn the light on in either game whenever a ball is in the plunger chute.

The Config Tool offers both options for the sake of flexibility, to let you customize it according to your tastes. If you want the button to light up for most games, even games that originally had plungers, choose Launch Button. If you want the button to faithfully simulate the way the real machines work, choose Authentic Launch Ball.

My advice is to choose according to whether or not you have a plunger in addition to the Launch button:

- If your cabinet *only* has a Launch Ball button, and no plunger, choose Launch Button. Since you don't have a plunger, you'll be using your Launch button in

every game, so it's nicer to have it light up most of the time.

- If your cabinet has *both* a plunger and a Launch button, use Authentic Launch Ball. That's better if you have both controls, because it will make your cabinet give more prominence to whichever control is appropriate for the game you're playing at any given time. When you're playing a plunger table, the Launch button will stay dark and discretely step into the background. When you're playing a plunger-less table, the Launch button will light up whenever a ball is ready, inviting the player's attention.
- If your cabinet only has a plunger and no Launch button, there's nothing to choose, so this whole question is moot!

ZB Launch Ball: This is an unusual output in that it's a "virtual" output only. You can ignore this for now, because it's not intended to be connected to a physical feedback device; it's related to the plunger setup instead. If you're curious, you can read about it in Chapter 109, ZB Launch Ball.

Fire Button: The lamp inside the Fire button, which is an extra button on top of the lockbar on some real machines. This is common on newer Stern machines, but various machines through the years have featured similar extra controls. It's uncommon on virtual cabinets, but some people include it for completeness. This light turns on in some games during special modes where the button becomes active. See Chapter 55, Button Lamps.



Extra Ball: The lamp inside the Extra Ball light. Many real machines from the 1990s had an "Extra Ball" or "Buy-In" button on the front panel, usually situated below the Start button or below the plunger. This let players buy an extra ball for a credit after the last ball drained. This lamp turns on in most games that had these buttons when the extra ball buy-in offer is available. See Chapter 55, Button Lamps.



10 Bumpers and 8 Bumpers: These are tactile feedback devices that simulate the mechanical kick of a pop bumper on a real machine. Virtual cab builders use a variety of devices to simulate these, including "contactors" (a type of electronic relay), automotive starter solenoids, open-frame solenoids, or even real pinball bumper coils. Most virtual cab builders set up an array of these devices spread across the middle and back of the playfield area so that the sound effects are properly positioned in space. See Chapter 59, Flippers, Bumpers, and Slingshots.

The names "10 Bumper" and "8 Bumper" are a little misleading. A "10 Bumper" setup actually has 6 bumper devices, and an "8 Bumper" setup really has 4 bumpers. It would be clearer if the names were something like "10 Solenoid" and "8 Solenoid", because the "10" and "8" don't count just the bumpers, but rather *all* of the solenoid-type devices in your whole system, including two for the slingshots and two for the flippers. So if you have two flippers, two slingshots, and four bumpers, you have an "8 Bumper" setup. If you have two flippers, two slingshots, and six bumpers, you have the "10 Bumper" setup.

Here's the typical "10 Bumper" device placement:



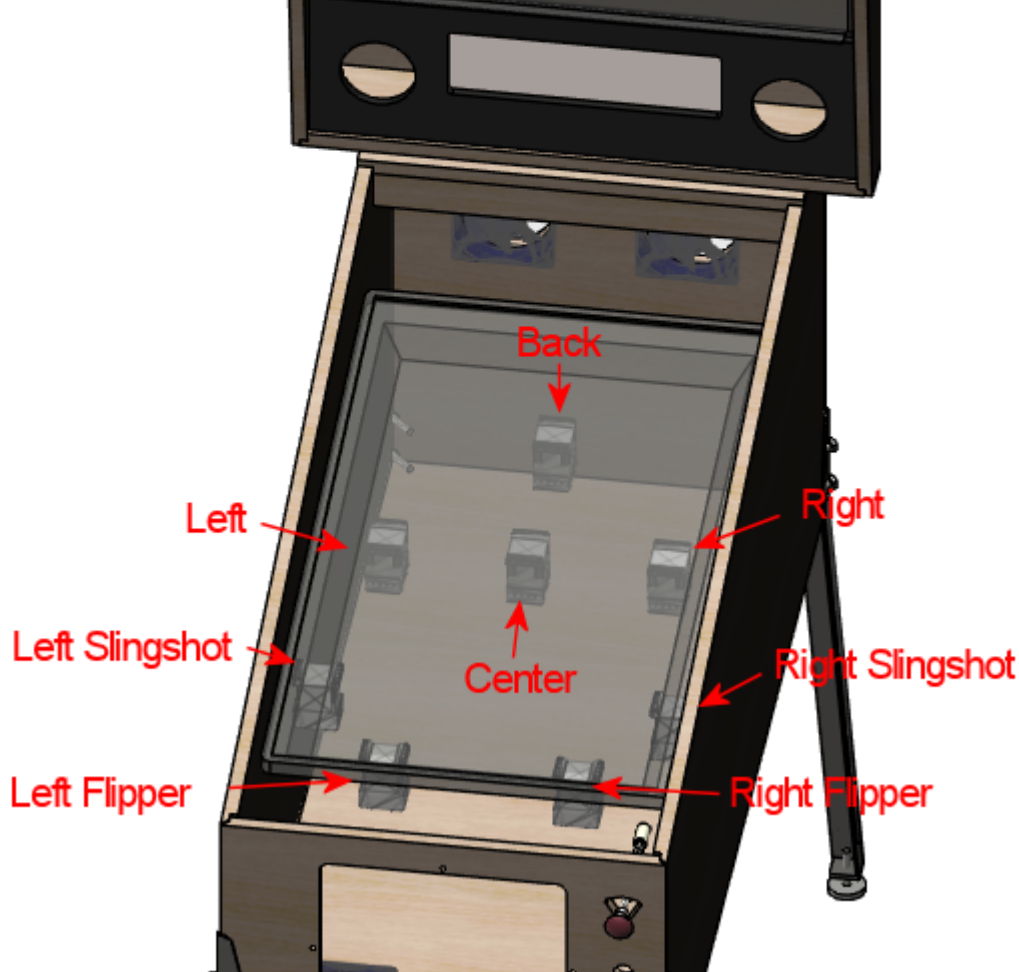


"10-Bumper" placement. The actual "bumper" simulators are the six devices across the middle and back playfield area. The two devices at the front are the flipper simulators, and the two just behind those are the slingshots.

If you're using ten of these solenoid-type devices overall, and you've arranged them roughly like this, you'd assign six of your ports as "10 Bumper" devices. The one at the back left corner is "10 Bumper Back Left", the rear center one is "10 Bumper Back Center", and so on for the others.

If you're building a smaller cabinet, or you just want to keep things a little simpler, the "8 Bumper" arrangement is almost the same, but only uses one device in the back row:





"8-Bumper" placement. The actual "bumper" simulators are the four devices across the middle and back playfield area. The flipper and slingshot devices are the same as in the "10-Bumper" layout.

Slingshot Left and **Slingshot Right:** The solenoid-type devices that simulate the mechanical kick of the slingshots on a real machine. The slingshots on a real machine are the kickers in the rubber-band triangles just above the flippers on most pinball tables. On a virtual cab, these are usually simulated by a pair of solenoid-type devices (the same types of devices used to simulate the bumpers - see above), usually placed near the front of the playfield TV on or near the left and right walls of the cabinet. See Chapter 59, Flippers, Bumpers, and Slingshots.



Flipper Left and **Flipper Right:** The solenoid-type devices that simulate the mechanical kick of the flippers. These are usually placed near the front of the playfield, set in slightly from the cabinet walls, to simulate the usual placement of flippers on real machines. See Chapter 59, Flippers, Bumpers, and Slingshots.

Knocker: A solenoid device that reproduces the replay knocker in a real cabinet. Many virtual cabs simply use actual pinball knockers for these. They're usually placed in the backbox, near the top left corner. See Chapter 60, Replay Knockers.

Shaker: A shaker motor. This is a motor placed inside the cabinet, with an off-balance weight attached to its shaft. When activated, it makes the cabinet vibrate, creating a mini-earthquake effect. Placement of the motor isn't critical, since the point is to make the whole cabinet shake, but it's usually fastened to the floor of the cabinet around the middle section. See Chapter 61, Shaker motors.

Gear: A gear motor, which is just what it sounds like: a small motor with gears attached to the shaft, placed somewhere inside the cabinet. The function in a virtual cabinet is purely as a sound effect device, to mimic of the sound of the small motors that are frequently used in real machines to animate playfield elements (think of the Thing hand coming out of its box in *The Addams Family*, the rotating trunk in *Theatre of Magic*, or the drawbridge in *Medieval Madness*). These are usually placed around the middle playfield, slightly rear of center, to allow for a reasonable approximation of a range of devices in real games. See Chapter 62, Gear motors.



Beacon: A rotating, lighted, police-car type beacon, usually placed on top of the backbox. This is a common backbox topper found on many real machines, and it's a favorite for virtual cab builders because it adds to the light show. See Chapter 57, Beacons.



Fan: A fan that blows air at the player when activated. A handful of real machines had these, notably *Whirlwind*, which had a 4" fan on top of the backbox that turned on during multiball and other game events. Virtual cab builders who include fans usually put them on top of the backbox, à la *Whirlwind*, but some people put them in other places, such as hiding them behind vents in the coin door. See Chapter 63, Fans.



Strobe: A bright white light, usually placed at the back of the main cabinet or on top of the backbox. Cab builders usually implement these using LED floodlights meant for use on pickup truck light bars. The strobe is essentially a virtual pinball community invention that doesn't correspond to any common feature from real machines, although a few real machines had something vaguely similar (notably *Flash*). See Chapter 56, Flashers and Strobes.

5 Flashers: The "5 Flashers" are a set of five separate bright lights, usually implemented with high-power, full-color "RGB" LED lamps. In real pinball machines, "flashers" were high-intensity lamps placed around the playfield in various spots to create dramatic lighting effects. Most pinballs from the mid 1980s had five or six of these placed strategically around the playfield, usually enclosed in colored plastic domes. Virtual cab builders can't easily place these around the playfield, since our playfields are actually TVs, so we usually position at the back of the main cabinet, or sometimes on top of the backbox. And we usually arrange them in a simple row of 3 or 5 evenly spaced lights.





Typical "5 Flashers" arrangement: five flasher domes in a row across the back wall of the cabinet above the playfield TV.

The Config Tool's "5 Flashers" devices are designed for cabinets that have a row of five of these lights. The Config Tool assumes that your flashers are arranged in a single horizontal row, so it gives you entries for the Outside Left, Left (which really means "Inside Left"), Center, Right (really "Inside Right"), and Outside Right positions.

The flashers are usually RGB devices, meaning that each one is actually made up of three separate LEDs - one Red, one Green, and one Blue. These three color LEDs have to be wired individually so that they can be controlled separate. That means that a single flasher actually acts like three separate devices on your output controller. You have to run three separate wires to three separate port connectors on your controller. For five flashers, you have to assign 15 ports.

The Config Tool knows all about RGB devices, but it imposes a special rule for them: the Red, Green, and Blue ports for each device **must** be consecutively numbered, and they **must** be in this order: Red, Green, Blue. For example, let's consider the Outside Left flasher. If you connect its Red LED to port 15 on your controller, then you have to connect its Green LED to port 16, and its Blue LED to port 17. The Config Tool doesn't give you any other way to group the color connections, so be sure to arrange your wiring in that order for each flasher device.

Once you've set up your port wiring in the correct Red-Green-Blue order, the Config Tool makes it really easy to set up an RGB device. You simply go to the port number where the Red LED of the group is connected, and you select, say, "5 Flasher Outside Left". This automatically assigns the whole group of three ports to the corresponding color channels for the same device. So even though you have to wire 15 physical channels to your 5 flashers, you only have to make five port assignments in the Config Tool - one for the Red channel for each flasher. The Green and Blue channels for each flasher will be automatically assigned to the adjacent channels.

See Chapter 56, Flashers and Strobes.

3 Flashers: If you're building a mini-cabinet, you might only have room for a row of three flasher lights. Choose the "3 Flashers" devices if you're using this configuration. Assign one set of RGB channels to each of your left, center, and right LEDs. As with the "5 Flashers", the DOF Config Tool requires these to be RGB devices, so you have to assign a block of three color channels (Red, Green, Blue) to each of the three flashers. See Chapter 56, Flashers and Strobes.

RGB Flippers: These are for miniature RGB LEDs inside the flipper buttons, to illuminate the buttons from within. The Config Tool assumes that you've wired your left and right flipper buttons together to the same output controller ports, so you just have to assign one set of outputs for these. As with the "5 Flashers", these are required to be RGB outputs, so a set of three consecutive ports is required, and they must be wired in Red, Green, Blue order. See Chapter 55, Button Lamps.

RGB Left Magnasave and RGB Right Magnasave: These are for miniature RGB LEDs inside your left and right MagnaSave buttons, which are the second set of flipper-like buttons found on some real machines and many virtual cabs. Like the flipper buttons, these can be illuminated from within with RGB LEDs. Unlike the

flipper buttons, DOF provides separate Left and Right channel assignments for these. The reason is that some real machines had asymmetrical MagnaSave buttons (such as only including a left or right button, or using different colored buttons on the two sides), so it's desirable to be able to control the two sides separately to replicate these asymmetries. As with the flipper button lights, these are required to be RGB devices, so you have to wire each Magnasave light to a block of three output controller ports, and arrange them in Red, Green, Blue order. See Chapter 55, Button Lamps.

RGB Undercab Smart and **RGB Undercab Complex**: These are for RGB light strips mounted on the underside of your cabinet, and sometimes the back of the backbox, to create a glowing pool of light around the machine. This is another case where there are two options for the same physical device type, to let you choose which type of programming you prefer. The "Smart" version generally uses a single, fixed lighting color for each game. The "Complex" version changes the colors in some games sync with game events, creating more of a light show. Some people prefer the light show effect, while others find it to be too distracting. See Chapter 58, Undercab Lighting.

"MX" devices: There are a bunch of entries in the list ending in "MX": PF Left Flashers MX, LF Left Effects MX, etc. You can ignore these when you're setting up your main feedback devices, because they're only for "addressable light strips", a special type of device that requires its own dedicated controller. See Chapter 65, Addressable Light Strips.

Coin: This is for the lamp inside your Coin button, if you have a separate button for this. A Coin button isn't something you'd find on real machines; it's a button that some virtual cab builders add so that they can easily simulate inserting a quarter. If you're using a standard door with coin slots, you could use the Coin output to control the lamps in the coin slots, but there's no reason to do this for the sake of authenticity: the coin chute lights on the real machines are simply wired to be permanently on. In any case, the Config Tool settings for the Coin lamp generally leave the Coin light turned on any time a table is loaded, so it's not much different from wiring the lamp to be always on. See Chapter 55, Button Lamps.

How to play: This is for the lamp inside another virtual-only button, this time a "How to play" button. This button is meant to display a game's instruction card when you're navigating an older menu system like HyperPin. This kind of extra button was fashionable for a while in the early days of virtual pinball, but it's more common now for cab builders to avoid buttons that aren't common on real pinball machines. And the newer menu systems like PinballX don't need dedicated buttons like this, since they expose extra features like this with on-screen menus instead of extra buttons. See Chapter 55, Button Lamps.

Genre: This is for yet another virtual-only button lamp, in this case a "Genre" button that lets you switch categories in older menu systems like HyperPin. Like the "How to play" button, this button isn't common on more modern virtual cabs, since the newer menu systems don't need it. See Chapter 55, Button Lamps.

Exit: The button lamp inside the Exit button. "Exit" is the one virtual-only button that you really can't do without; it stops the current game and returns to the menu system. You need a special button for this function because real pinball games simply don't have a concept of "exiting" to a menu system. See Chapter 55, Button Lamps.

Custom Output 1-4: These are extra outputs available for you to assign to any unique feedback devices in your cabinet that don't correspond to anything else in the Config Tool lineup. The standard Config Tool database doesn't do anything at all with these outputs, but it lets you define your own rules for triggering them during game

play.

Be warned that it can be a lot of work to set these up, because you have to manually create rules for them in every table where you want to use them. See Appendix 7, Customizing a table's DOF effects for details.

Custom RGB 1-2: These are just like **Custom Output 1-4** above, but in this case they're for your custom RGB lighting devices. That is, devices with color channels for Red, Green, and Blue.

Bell: A large mechanical bell, such as the one on the top of *Fire*'s backbox, which some games use for dramatic effect. This type of bell is meant to produce a loud, deep note like a church bell. See Chapter 64, Chimes and Bells.

Chime Unit High Tone through **Chime Unit Low Tone:** These are for a traditional chime unit with three chime bars, of the sort used in many electromechanical pinballs of the 1960s. You can install a real chime unit (or a replica) for more authentic re-creations of the sound effects in older games. See Chapter 64, Chimes and Bells.

Chime Unit Extra-Low Tone: A fourth chime bar, if you have a four-bar chime unit like those used in some Bally machines in the late 1970s. See Chapter 64, Chimes and Bells.

Chime 5: This is for a fifth chime bar, which (as far as anyone can tell) is a DOF extension that doesn't correspond to real chimes in any real machines, but can be used to add extra variety to EM game play by adding a fifth chime tone. See Chapter 64, Chimes and Bells.

Shell Bell Small and **Shell Bell Large:** These are for a pair of shell bells, which are similar to chime units but use circular ringing elements for a different tonal effect. Some EM machines from the 1960s and 1970s used this instead of chime bars. See Chapter 64, Chimes and Bells.

Repeating Bell: A bell that mechanically strikes repeatedly as long as it's energized, like the bell in an old-fashioned telephone or in a fire alarm. A few games (*Space Shuttle*, *Taxi*) use repeating bells for sound effects during play. See Chapter 64, Chimes and Bells.

Hellball Motor and **Hellball RGB:** These outputs are designed to control a Varytec Hellball, which is a sort of disco party ball. The Motor output controls motion, the RGB output controls the spotlight color.

Appendix 6. DOF Event Codes

How does Visual Pinball know which flashers to fire when the ball hits a target, or when to turn on the shaker motor?

VP *doesn't* know, really; VP just thinks in terms of events related to the game, such as a certain target being hit or a certain simulated lamp on the playfield lighting up. VP passes those events to DOF, telling DOF, for example, "switch 15 on the playfield was triggered". DOF is the part where these game-specific events are converted into feedback device actions in your cabinet, such as a flasher firing yellow or the shaker motor activating.

But that still doesn't answer the question! It just changes it from *how does VP know?* to *how does **DOF** know?*

DOF knows because the configuration files tell it so. These are the .ini files that you download from the DOF config tool when you click the "Generate Config" button, and these files in turn get the information from the gigantic list of cryptic codes in the Table Configs section of the config tool - the lines that say things like "L57 m550 Blink fu500 fd550".



Those codes are painstakingly hand-crafted for each individual table, to create a pleasing set of effects for each game during play. Thank goodness (or more to the point, thank Arnggrim) that someone (Arnggrim) *already* hand-crafted a good set of codes for nearly all of the tables in existence, and put them in the DOF Config Tool's public database for all of us to use. That saves you the work of going through that whole coding process for the many dozens of tables you'll eventually want to install on your cab.

If you're happy with the default DOF effects provided by the Config Tool's database, and you don't want to make any customizations of your own, then you don't have to know anything about how DOF works apart from the basics of installing it on your system, which we covered separately in Chapter 46, DOF Setup. But if you do want to customize anything, you'll have to learn what the cryptic DOF codes mean. DOF's documentation has some of this information, but it's incomplete and (in my opinion) poorly organized and almost as cryptic as the codes themselves. So this section is my attempt at a more approachable and more thorough explanation of what the codes mean and how to use them.

The actual procedure for customizing tables in the Config Tool is covered separately, in Appendix 7, Customizing a table's DOF effects. The present section covers just the syntax that you enter in the Config Tool to make those customizations.

Event/Toy/Port associations

Before we get into the event code syntax, it might be helpful to explain the overall conceptual framework that the Config Tool uses. The elements of this framework aren't what I'd call intuitive, so I think it's worth taking a moment to think about the "data model".

The real function of the DOF Config Tool is to tie together two big collections of data. Each collection is like a "database table" or a spreadsheet, where you have a list of information organized into rows and columns. Here are the big things that the Config Tool keeps track of for you:

- The port assignments. This is where you tell the config tool which "toy" is attached to which numbered output port on your output controllers. In DOF speak, a "toy" is a particular feedback device that carries out some concrete effect, such as the shaker motor or the left rear bumper solenoid. For example, you might tell the Port Assignments list that Port 17 on your LedWiz is the shaker motor.
- The Table Config for each game. This is a big list of the event codes assigned to each toy. This relates the abstract simulation events in Visual Pinball to one or more concrete effects on a given toy, on a game-by-game basis. For example, if you go to the table config for *Funhouse*, the Shaker Motor line might have codes that say "run for five seconds whenever switch 19 is activated". ("Switch 19" is the part that Visual Pinball knows about, and it represents some action on some game element, such as a particular target getting hit by the ball.) If you go to the table config for *Earthshaker*, the Shaker Motor line will have a whole different set of triggers appropriate to that game.

The Config Tool's main job is to take those two lists, and combine them into the *even more cryptic format* that the DOF .ini files use. When you click Generate Config, the Config Tool sorts through all of the elements in those lists to form the hardware assignments for your machine. It combines "run the shaker motor for five seconds when switch 19 activates" and "shaker motor is LedWiz port 17" to form "turn on LedWiz port 17 for five seconds when switch 19 activates". That's how DOF knows to turn on the shaker motor in your specific cabinet wiring at the right time in the game.

It's a lot of abstraction to keep track of! But it makes sense when you consider that everyone's cab is unique, and every pinball table is unique. It would be terrible if the DOF just assumed that "Port 17" meant "Shaker Motor", or that "Switch 19" is the "left drop target". Those numbers might be right on one person's cab and one Visual Pinball table, but they're certainly not universal. So we really do need the multiple levels of abstraction. At least we have the Config Tool to make these connections for us, so that we don't have to put this all together in the right order by hand.

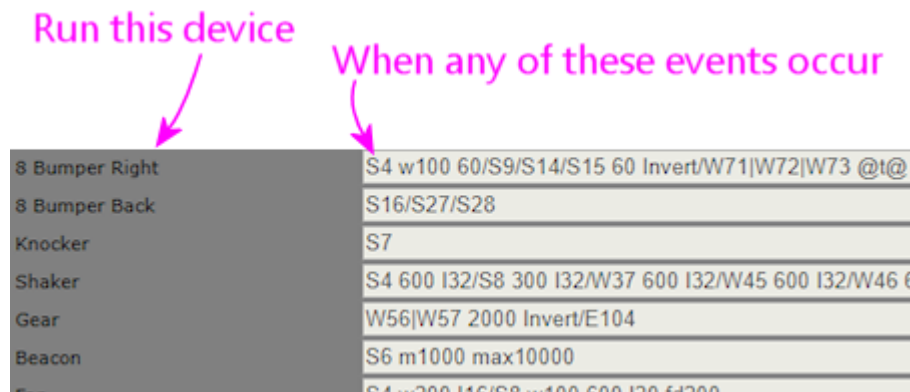
Toy/event mappings

Since the DOF Config Tool takes care of the details of mapping everything to your output controller and physical hardware ports, the only thing we have to worry about when customizing an effect is the event codes.

Whenever I think about an effect I want out of DOF, my intuition is to think about it like this: "when *this thing happens in the game*, I want *this toy to fire*". For example, I might say "when the ball hits the castle, I want the shaker to fire for a couple of seconds".

To get into the Config Tool's (and DOF's) way of thinking, you have to turn that around: "The shaker fires for a couple of seconds when the ball hits the castle."

That might sound like the same thing, and it really is, but the ordering of the nouns is important. The reason you have to think about it as "Toy fires when Event happens" is that the entries in the Table Configs page are all organized that way:



8 Bumper Right	S4 w100 60/S9/S14/S15 60 Invert/W71 W72 W73 @t@
8 Bumper Back	S16/S27/S28
Knocker	S7
Shaker	S4 600 I32/S8 300 I32/W37 600 I32/W45 600 I32/W46 600 I32/E102 600 I32
Gear	W56 W57 2000 Invert/E104
Beacon	S6 m1000 max10000
	S4 ...300 I45/S8 ...400 600 I32 64300

This becomes more apparent when you want to do something like "When I hit the Extra Ball target, run the shaker motor for two seconds and flash red on the middle flasher." For that case, you have to turn it into two commands: "Shaker runs for two seconds when I hit extra ball", *and* "Middle flasher flashes red when I hit extra ball". You have to break it into two pieces like that because you're going to have to enter the trigger event twice, once under Shaker Motor and once under Middle Flasher.

Events

Now we come to the part I really wanted to document here, because it's just not written down anywhere else as far as I can tell: what do those event codes in the cryptic strings on the right side mean?

There are lots of details to get into, but at a high level, the codes are all about (a) specifying some physical object in the table that makes the device activate, and (b) specifying the timing details about when and for how long it activates.

Let's look at an example, and take it apart to see how it works. We'll use the shaker motor (the "Shaker" row on the Table Configs page) for *Medieval Madness*. Here's the default definition in the DOF database:

```
S4 600 I32/S8 300 I32/W37 600 I32/W45 600 I32/W46 600 I32/E102 600 I32
```

The first step in deconstructing this is to split things up at the "/" marks. Those let us combine independent triggers, to give the shaker motor more than one triggering game event. So what we really have here is six separate, independent triggers for the shaker motor:

```
S4 600 I32
S8 300 I32
W37 600 I32
W45 600 I32
W46 600 I32
E102 600 I32
```

What we mean by "independent" is that all of these triggers are always in effect, without any interaction with one another. If any one of the triggers occur, DOF will fire the shaker. They don't interact with each other in any way, so we can look at

each one separately, as though the others didn't exist.

Within each trigger, the first item is always the triggering event. Most triggering events are simulated objects in the table, such as:

- A lamp (e.g., an Extra Ball When Lit light) turning on
- A solenoid (e.g., the top bumper or the left slingshot) firing
- A switch (e.g., the rollover switch in the right outlane) closing

DOF represents most of these game objects with one letter codes: S is for Solenoid, W is for sWitch, L is for Lamp. There's a full list later in this section. There's also a number after the code indicating which specific switch/lamp/whatever we're talking about: S8 in the first line is Solenoid 8. Figuring out the number that corresponds to a given solenoid/lamp/whatever in a given game is a whole separate (and equally cryptic) subject that we'll get into in Appendix 7, Customizing a table's DOF effects.

So the first event, S8 300 I32, means that we fire this effect when Solenoid 8 in the game turns on. The second one, W37 600 I32, fires when sWitch 37 closes. And so on.

What does "fire this effect" mean, exactly? To a first approximation, it just means "turn on this device". For example, if we had a trigger in here that **only** said L11 and nothing else, it would mean "turn on the shaker motor whenever lamp 11 is lit, and keep it on the whole time it's lit." Most triggers aren't quite that simple, though. The ones we're looking at for the shaker all have some extra gibberish at the end, which modifies the simple "turn the device on when this table object is on" and makes it a little more subtle.

Each of the triggers in this example has the same format: *trigger-object number Inumber*. This is a really common format that you'll see all over the place in the DOF config. When you see a bare number like that immediately after the trigger object, it sets the duration for the effect - how long the effect runs. The value is always in milliseconds. So S8 300 means "run the shaker motor for 300 milliseconds when Solenoid 8 fires". This is an important distinction from just S8, which would mean "run it for exactly as long as Solenoid 8 fires".

There's also a way of setting a minimum and/or maximum duration for the effect instead of setting an exact time for it. S8 M50 means "run for at least 50 milliseconds, but keep going longer if Solenoid 8 stays on longer", and S8 MAX300 means "run for as long as Solenoid 8 stays on, but stop after 300 milliseconds no matter what". You can combine min and max times, too, as in S8 M50 MAX300. None of the *Medieval Madness* shaker events use the min/max limits, but you'll see them in other events.

The final item in all of these definitions is I32. "I" is another modifier meaning "run with this intensity". The intensity is on the rather odd scale of 0-48 (why? because DOF was originally designed around the LedWiz, and the LedWiz uses a 0-48 brightness scale). So I32 is about 2/3 of full intensity.

There's another common detail that's worth calling out. Some devices are "RGB" devices, meaning that they're lighting devices capable of showing different colors by blending red, green, and blue light at different brightness levels. In your physical output controller wiring, of course, this has to be wired as three separate ports, one for each color channel. The Config Tool mercifully combines the three channels into one line item, so you'll see "5 Flasher Left" as a single device rather than as its three physical wiring channels. For these RGB devices, the Config Tool lets you set a color as part of the programming, so you might see things like S12 Magenta. That means that we set the RGB channel mixing to show Magenta when Solenoid 12 activates.

You can also use HTML-style #rrggbbaa syntax, with hex numbers for red, green, blue, and alpha (transparency, usually just set to FF for fully opaque). For example, #ff0000ff is 100% red.

Now that you know how to take these definitions apart, you still need a list of all of the specific code letters. The rest of this section is basically that.

Table variables

Before we get to the cryptic strings, there's actually one weird special case that we have to mention first. If you open up the DOF Config Tool to the Table Configs list, you'll see that a slot at the top of the list called "Table Variables".

That slot is special. It's not like the others. It looks like the others, but it's not the same thing at all.

Table Variables is a unique slot where you can enter symbolic names for more complex expressions that you want to use elsewhere in the effects list. These are essentially macros, if you're familiar with that term: text that will be substituted for the variable name.

To use a table variable, you surround the variable name with "@" signs. For example, to use the variable playon, you'd write @playon@.

Variable names are defined like this in the Table Variables box:

```
playon=(W43=0)
```

This means that the variable **@playon@** will be replaced by the text (W43=0) wherever it appears in a toy definition line. Note that the parentheses are part of the substituted text.

Pre-defined global variables

In addition to the table variables, there are some pre-defined variables that are always available. As with the table variables, these are simply substituted into the text where they appear.

Variable	Description
@dt@	The "drop targets" settings from your Port Assignments page, in the form <i>duration Intensity</i> . For example, if you have the default 60ms duration and 48 intensity, @dt@ will expand to 60 I48.
@t@	The "targets" settings from your Port Assignments page, in the form <i>duration Intensity</i> . For example, if you have the default 60ms duration and 48 intensity, @t@ will expand to 60 I48.

@allrgb@	Applies only to RGB devices (flashers, RGB flipper buttons, etc), and works only for certain pseudo-tables that represent "front ends" and other special programs, such as PinballX and PinballY. Substitutes the RGB colors for the device where the rule appears, for the "current table", whatever that means in the program context (for PinballX and PinballY, it's the game selected in the wheel UI). For example, if you use @allrgb@ within the RGB Left Flipper rule, the RGB Left Flipper color for the "current table" is used. (In order for this to work, the client program must activate a named DOF event using the DOF ROM name of its current table selection at any given time. PinballX and PinballY do this automatically.)
----------	---

Multiple event triggers

Each effect slot can have any number of independent effects, separated by slashes ("/"):

S16/S27/S28

That means that this toy should be activated on *any* of the listed events - S16, S27, or S28.

Event syntax

Within the "/" elements, an event looks like this:

trigger-code effect-codes

The trigger code specifies *when* this effect is fired; the effect codes specify *what* happens. So the way you read this is "When *trigger-code* happens, do *effect-codes* on the current device". The trigger codes are mostly things that happen in the game, such as "switch 9 hit", and the effect codes are mostly things like "fade in for 100 milliseconds". So we can put this together to say things like "when you hit the castle (switch 9), fade this light on for 250ms then fade it back out for 100ms".

You can list multiple trigger codes for the same effect, by separating them with vertical bars ("|"), so you could have something like this:

S7|S9 fu100 fd250

That means that we trigger this effect on S7 *or* S9, and carry out the effect codes "fu100 fd250".

Trigger codes

The first element of an event code is the "trigger", which specifies what event in the game makes the toy activate. Remember the multiple trigger codes can be combined with "|".

Code

Example

Description

<i>\$name</i>	\$PBYMenu	Named event. These are pseudo-events defined by the table, so they're specific to the table. I think these are mostly (only?) used for non-pinball programs such as PinballX and PinballY, which obviously don't have actual table events (switches and solenoids and so on) to work with. The PinballY documentation has a full list of PinballY's codes.
<i>(condition)</i>	(S7 > 1)	A condition event. The toy fires whenever the condition evaluates to true, or a non-zero integer value. See below for more.
0	0	Same as OFF.
1	1	Same as ON.
<i>Bnumber</i>	B1	Score digit. I don't know how these are used; I think they're related to scoring reels in EM games.
BLINK	BLINK	The toy is always on, and blinks on and off at 1-second intervals.
<i>Cnumber</i>	C1	Score. I don't know how these are used; I think they're related to scoring reels in EM games.
<i>Dnumber</i>	D4	LED. Refers to a controlled LED in a 2000s Stern game. The numbering is determined by the original game ROM programming.
<i>Enumber</i>	E4	<p>EM table element. The "E" events are programmed in a table's Visual Basic scripts, specifically to trigger DOF effects. The event numbers are arbitrary and up to the table author. If you look at a table script for an EM table that's been programmed with DOF effects, you'll find lines like this:</p> <pre> Controller.B2SSetData 11,1 DOF 118,1 </pre> <p>The first number in the pair in these lines is the "E" event code. The second number is the value for the event, usually 0 for OFF and 1 for ON. So "E4" in the DOF config is triggered when the table script executes a line like DOF 4,1.</p>

<i>Gnumber</i>	G1	General illumination (GI) string. The toy is activated when the given general illumination on the playfield is switched on. GI refers to the little lamps scattered around the playfield, mostly under the plastics, that provide background lighting. These lights aren't individually controlled; they're controlled as a group, known as a GI "string". Some games only have one GI string for all of the playfield lights, and some divide the GI lighting into two or more strings. Many of the 1980s and 1990s games have separate GI strings for the upper and lower playfield lights. The exact layout is determined by the game's original programming.
<i>Lnumber</i>	L7	Lamp. The toy is activated when the given lamp on the table is on. Lamps refer to the individually controlled lights on the playfield or backbox, such as an "Extra Ball When Lit" light or a bumper lamp. The lamp numbers are defined by the particular table, according to the game's original ROM programming. See Appendix 7, Customizing a table's DOF effects for tips on figuring out which lamp is which for a given table.
<i>Mnumber</i>	M3	Mechanical object. The toy activates when the given mech object is activated. Mech objects are special programming added to a few games in VPinMAME or B2SServer, not something from the original real table. I don't know of a list of these objects anywhere; I think you just have to look at the VPinMAME source code to find out about them.
<i>Nnumber</i>	N3	Mech object value from GetMech. I have no idea what this is about.
OFF	OFF	Another pseudo-event, meaning that the toy is always off.
ON	ON	A pseudo-event meaning that the effect is always on when the table is running. This is typically used for things like the Coin and Exit buttons that you just want to stay lit all the time.
<i>Snumber</i>	S15	Solenoid. The toy is activated when the given playfield solenoid is activated. Solenoids refer to (simulated) physical solenoids in the game, such as bumper coils, kick-out coils, and slingshots. The Williams games also controlled the flasher lamps as though they were solenoids (since they required relatively high power), so some solenoid numbers refer to flasher lamps on original playfields. The solenoid numbers are assigned by the game's original ROM programming; see Appendix 7, Customizing a table's DOF effects.

<i>Wnumber</i>	W9	Switch. This toy is activated when the given playfield switch is activated. A switch is usually a (simulated) physical switch on the playfield, like a rollover switch, stand-up target, or bumper contact. The switch numbering is determined by the game's original ROM programming; see Appendix 7, Customizing a table's DOF effects.
----------------	----	---

Effect codes

The part following the "trigger code" is a list of effect codes, separated by spaces.

Code	Example	Description
<i>number</i>	100	<p>A number on its own sets one of two things:</p> <ul style="list-style-type: none"> • If there's a BLINK specifier before it, this is the blink interval, in milliseconds • If there isn't a BLINK specifier, this sets the duration of the effect, in milliseconds; the effect will run for exactly this long, even if the trigger turns off earlier or stays on longer
<i>number number</i>	100 300	If there's a BLINK specifier, a pair of numbers sets the blink time and overall effect duration, both in milliseconds. (This is invalid syntax in the absence of a BLINK command.)
<i>color</i>	red or #FF0000FF	This can only be used as the first effect code in the list, and can only be used for RGB devices like flasher LEDs, flipper button lights, or undercab light strips. This specifies the color to use for the effect. This can be a color name taken from the pre-defined set in the config tool (listed at the bottom of the Table Configs page), or it can be an HTML-style #rrggbbaa code, with hex values for <i>rr</i> (red), <i>gg</i> (green), <i>bb</i> (blue), and <i>aa</i> (alpha transparency channel, almost always FF for fully opaque).
BL# <i>low</i>	BL10	Sets the brightness of the "low" part of the blink cycle, from 0 (fully off) to 255 (fully on). By default, this is 0 for fully off.
BL/ <i>low</i>	BL20	Sets the "low" blink brightness on scale of 0 to 48. This is exactly the same as BL# except for the different scale used. BL48 is the same as BL#255. (This older scale is an historical relic from when DOF was all about LedWiz access, as the LedWiz uses a native 0-48 brightness scale. DOF generalized this to a 0-255 scale for finer shades of brightness possible on more modern controllers.)

BLINK	BLINK	Blinks the effect on and off at 1-second intervals
BNPtime	BNP20	Sets the "nested" blink interval, in milliseconds.
BNPWpct	BNP25	Sets the "nested" blink pulse width, as a percentage of the nested blink interval (1 to 99). For example, BNP25 sets the ON time of the blink to 25% of each blink interval.
BPWpct	BPW25	Sets the blink pulse width, as a percentage of the blink interval (1 to 99).
Etime	E20	Extended duration, in millisecond. This makes the event continue for the given duration after the trigger turns off.
Ftime	F100	Sets the fade in/out duration for the effect, in milliseconds. This sets the fade-in and fade-out to the same interval; they can also be set separately via FU and FD.
FDtime	FD100	Sets the fade-out (-down) duration for the effect, in milliseconds.
FUtime	FU100	Sets the fade-in (-up) duration for the effect, in milliseconds.
I#intensity	I20	Sets the intensity/brightness of the effect, from 0 (fully off) to 255 (fully on).
Iintensity	I20	Sets the intensity/brightness of the effect, from 0 (fully off) to 48 (fully on). This is the same as I#, but uses the older/coarser LedWiz 0-48 scale.
INVERT	INVERT	Inverts the trigger. For example, S7 INVERT means that the event is triggered when Solenoid 7 is off rather than when it's on.
Llayer	L3	Sets the "layer" of the effect. Layers allow multiple effects to overlap in time.
Mduration	M60	Sets the minimum duration of the effect, in milliseconds. Normally, the effect ends as soon as the trigger turns off.
MAXduration	MAX500	Sets the maximum duration of the effect, in milliseconds. The effect ends after this time even if the trigger condition remains activated. Normally, the effect continues for as long as the trigger remains on.

NOBOOL	NOBOOL	Makes the trigger non-boolean. The trigger codes (e.g., S7 for Solenoid 7) are all integer values from 0 to 255 in the interface between Visual Pinball, VPinMAME, and DOF. By default, DOF reinterprets these as simple OFF or ON values, by considering 0 to be OFF and everything non-zero to be ON. So 1 is ON, 2 is ON, etc. NOBOOL overrides this and makes DOF apply the trigger's actual numeric value as the brightness/intensity level for the effect. This can be used for tables where the ROM software controls the brightness of a flasher, for example, to pass the flasher brightness value through from the ROM to the output device.
<i>Wtime</i>	W50	Sets the wait time in milliseconds. This is the wait time after the event trigger turns on before the effect starts (normally zero, so that the effect starts immediately when the trigger occurs).

The following all apply to "area effects", for addressable light strip matrices.

Code	Example	Description
<i>AAaccel</i>	AA10	Sets the area effect acceleration.
<i>AABbehavior</i>	AABL	Sets the area bitmap animation frame repetition behavior (O = play once, L = start at first frame then loop, C = start at next frame from last playback then loop)
<i>AACcount</i>	AAC10	Sets the area bitmap animation step count.
<i>AADdir</i>	AADF	Sets the area bitmap animation frame direction (F = step by frame in animated GIF, R = step from left to right through source image, D = step from top to bottom through source image)
<i>AAFduration</i>	AAF3	Sets the area bitmap animation frame duration.
<i>AASstep</i>	AAS5	Sets the area bitmap animation step size.
<i>ABFframe</i>	ABF3	Sets the area bitmap frame.
<i>ABLleft</i>	ABL5	Sets the area bitmap left.
<i>ABHheight</i>	ABH10	Sets the area bitmap height.
<i>ABTtop</i>	ABT5	Sets the area bitmap top.
<i>ABWwidth</i>	ABW10	Sets the area bitmap width.

<i>ADshift</i>	ADL	Sets the area shift direction (L=left, R=right, U=up, D=down).
<i>AFDENDensity</i>	AFDEN5	Sets the area animation flicker density.
<i>AFMAXduration</i>	AFMAX60	Sets the maximum area animation flicker duration in milliseconds.
<i>AFMINduration</i>	AFMIN7	Sets the minimum area animation flicker duration in milliseconds.
<i>AFFADEduration</i>	AFFADE100	Sets the area animation flicker fade time in milliseconds.
<i>AHheight</i>	AH20	Sets the height of the matrix effect area.
<i>ALleft</i>	AL0	Sets the left of the matrix effect area.
<i>APCcolor</i>	apcRED	Sets the color for a plasma effect; see www.bidouille.org/prog/plasma .
<i>APDdensity</i>	APD25	Sets the density for a plasma effect, 0 to 100; see www.bidouille.org/prog/plasma .
<i>APSspeed</i>	APS10	Sets the speed for a plasma effect; see www.bidouille.org/prog/plasma .
<i>ASspeed</i>	AS5	Sets the area speed.
<i>ASAaccel</i>	ASA10	Same as AA.
<i>ASDshift</i>	ASDU	Same as AD.
<i>ASSspeed</i>	ASS5	Same as AS.
<i>ASSspeedMS</i>	ASS100MS	Sets the area speed in milliseconds.
<i>ATtop</i>	AT0	Sets the top of the matrix effect area.
<i>AWwidth</i>	AW20	Sets the width of the matrix effect area.
<i>SHPshape</i>	SHPNumber0	Shows the given pre-defined shape. The shapes are defined in a separate DOF config file, DirectOutputShapes.xml , which is included in the ZIP file that the DOF Config Tool downloads when you click Generate Config. You can look through that file for a list of available shapes.

Conditions

If a trigger code is enclosed in parentheses ("()"), it's a condition expression.

For the most part, these are used for simple And/Or combinations of multiple event triggers. For example, (w7 And L9) is a combined trigger that fires when Switch 7 and Lamp 9 are both on, and (s5 or s6) fires when either Solenoid 5 or Solenoid 6 is on.

You can create more complex conditions than that, though. The parser uses a full

expression language, based on C# notation. All of the trigger codes can be used as variables ("S7" for solenoid 7, for example). They evaluate to numeric values based on the values sent from the VPInMAME or the table's Visual Basic scripts; in most cases, these will simply be 0 for off and 1 for on, but they can take on other values in some cases, usually limited to a range of 0 to 255. EM table simulations sometimes use the "E" event codes with numeric values, for example.

The following operators are available:

Operator	Description	Example
+	Add	100 + S7
-	Subtract	100 - S7
*	Multiply	100 * S7
/	Divide	S7 / 100
%	Remainder	S7 % 100
^	Power	S7 ^ 3
-	Negation	-S7
+	Concatenation	"abc" + S7
<<	Left bit-shift	S7 << 3
>>	Right bit-shift	S7 >> 3
=, ><, <, >, <-, >=	Comparison	S7 > 100
And, Or, Xor, Not	Boolean logic	(\$x > 1) and (S7 < 10)
And, Or, Xor, Not	Bitwise logic	S7 and 0x0F
If	Conditional	If(S7 > 1, "yes", "no")
Cast	Type conversion	Cast(S7, int)
[]	Array index	S7[1]
.	Property/member	S7.color

You can also use the following literal value types:

Description	Example
String	"red"
Character	'c'
Boolean	true, false
Real (double)	100.25

Float	100.25f
Integer	100
Integer (unsigned)	100U
Integer (64-bit)	100L
Integer (64-bit unsigned)	100LU
Hex integer	0xFF, 0x1000L, 0xFFFFU

Appendix 7. Customizing a table's DOF effects

This section covers the gnarly details of DOF and VPinMAME that you need to know if you want to customize a table's DOF effects. This is useful if:

- You have unique feedback devices in your cabinet that most tables don't know about, and you want to trigger them on certain game events in certain tables
- You want to add some new DOF effects to an existing table
- You want to extra DOF effects to an existing table that doesn't already have any DOF support
- You're creating your own VP tables from scratch, and you want them to support DOF feedback effects

To make use of the information in this section, you also have to understand how DOF effects work. That's described in the previous appendix, Appendix 6, DOF Event Codes. That section explains how to create the effects you want for a given event; this section is about how to identify those source events in a given game.



Advanced topic warning! You don't need to know any of this to use DOF with the standard configuration settings. The details are rather complex and ugly. If you're setting up your cabinet for the first time, I'd recommend filing a mental note that this section is here and skipping ahead. You can always come back to it later.

How to add a new table to the DOF config tool

If you're creating your own original table, or a re-creation of a real table that no one has implemented in VP before, you'll want an entry for the table in your DOF config files. And assuming you're generating your files with the DOF config tool, that means adding an entry to the online database.

So how do you add a new table to the online database? Well, you can't - not directly, at least. There's no provision for users to add new tables, even private ones. Only the site administrators can do that. Fortunately, said administrators' attitude is that every table under the sun should be included in the master list, and they've always been quick to respond when I've suggested missing tables. So the way to get a new table added is to request it from the site admins. The best way I've found to reach them is to post the request to the following forum on VPUniverse: Digital Pinball Cabinets > Direct Output Framework > Direct Output Support.

How to customize an existing table

If the table you want to customize is already in the DOF Config Tool's online database, it's easy to add your own customizations:

- Open the DOF Config Tool in your browser
- Click the Table Configs tab
- Select the table you want to customize from the drop list at the top
- The left column shows the "Public Configuration", which is the default effects defined in the DOF database
- The right column shows the "Candidate Configuration", which is where you make your custom changes. By default, those boxes are all set to copies of the public config. You can change any effects you want to by typing your own

effects codes into the boxes on the right side.

- The effects codes are entered per device. The way you have to think about it is "I want the shaker motor to run when (some game event) occurs". To make that happen, you find "Shaker" in the list of devices, and you enter the event codes representing the triggering game event in the Candidate Configuration box on the right side next to Shaker.
- The DOF trigger codes are extremely cryptic, but we've tried to provide a full accounting of them in Appendix 6, DOF Event Codes. Read that section to figure out what to enter in the box.
- After you've made your changes, click Save Changes at the bottom
- Click Generate Config at the bottom to download the new DOF .ini files; when they finish downloading, extract them into your DOF install folder, replacing the existing files
- You'll have to exit completely out of VP or any other programs before testing the updates (exit out of the whole program, not just "Play" mode)

Example: Disable the flipper solenoid effects in PinballX or PinballY

Here's a quick example of how to use the customization procedure to accomplish a request I've seen several times on the forums: "I want to get rid of the flipper solenoid effects in my front end program, but I want to keep the flasher effects. How do I disable just the flipper effects?"

PinballX and PinballY both accomplish their DOF effects using the normal DOF configuration, by pretending that they're pinball tables named "PinballX" and "PinballY" (respectively). This means that you can customize the DOF effects for the front-end programs as though they were pinball tables, using the same techniques described in the rest of this section. This is a particularly easy special case, though, since all we have to do is delete the flipper solenoid triggers. So let's walk through the step-by-step procedure:

- Open the DOF config tool in your browser
- Log in
- Click the **Table Configs** tab in the top navigation bar
- In the **Table Name** drop list, select your preferred front-end program (**PinballX** or **PinballY**)
- Find the **Flipper Left** and **Flipper Right** rows in the list of toys. These rows specify the program events that will cause these solenoids to fire while you're running the front-end program.
- In the right column of each row, simply delete all of the text in the edit box, so that the box is complete empty. This removes all of the events that would cause the flipper solenoids to fire while you're running the front-end program, which will have the effect is disabling the solenoid effects entirely. (The changes we're making here will **only** affect the table that we selected in the **Table Name** drop list earlier - in this case, your front-end program. They won't affect or disable the solenoid effects in any other games.)
- At the bottom of the page, click **Save Changes**
- At the bottom of the page, click **Generate Config**, and wait for the browser to download the ZIP file containing your new configuration files
- Unzip the **.ini** files in the downloaded ZIP, and use them to replace the old ones in the **Config** folder within your DOF install folder

You can use this same procedure to add your own custom DOF effects to the front-end programs. Disabling the flippers is just a really easy special case, because all you have to do is delete the DOF instructions that activate the solenoids.

EM (Electro-Mechanical) games

If the table you're customizing is from the 1970s or earlier, and used mechanical score reels instead of software, its handling in DOF is completely tied to the Visual Basic script for the game.

EM machines didn't have any software, so VPinMAME isn't involved. The table script (written in Visual Basic, assuming we're talking about VP) contains all of the game's logic, including all of the scoring rules and sound effects. And including DOF effects. DOF keys everything in these games to "event codes" written into the Visual Basic scripts.

If you look at one of those tables in the DOF database, you'll see that everything is keyed to "E" codes - E17, E93, etc. Those "E" codes correspond to explicit DOF calls in the Visual Basic script for the table. They usually take one of these forms:

```
Controller.B2SSetData 11,1
DOF 118,1
```

To customize DOF events for an EM table, then, you need look no further than the table script:

- Launch VP
- Open the table in the editor
- On the menu, select **Table > Script** in VP 10 or **View >: Script** in VP 9

Scan through the script for lines like the above, with calls to B2SSetData and DOF. The first number in each call like that corresponds to an "E" code in the DOF Config Tool trigger. For example, DOF 118,1 means "set event code E118 to value 1", so a Config Tool trigger of E118 would fire when that Visual Basic script executes.

Note that the second number in each B2SSetData or DOF command is the value to set for the event code, so DOF 118,0 turns the event off. Most "E" codes are only triggered momentarily, because they're intended to fire some effect in response to something happening in the game.

ROM-based games

ROM-based games are the ones from the solid-state era, when they started using little 8-bit computers inside the backbox to do the scoring, music, etc. Those computers ran software burned into a ROM chip (read-only memory). It's like the cartridge in an old Atari video game console.

Visual Pinball handles ROM-based games using VPinMAME, which runs the original ROM software in emulation. The Visual Basic scripts in these games don't have to do any of the scoring or music playback, and they don't have to figure out when to fire the kickers or light the playfield lamps. That's all handled by the original ROM software. VPinMAME runs the ROM and sends commands to VP to tell it when the ROM wants to fire a kicker or light up a lamp.

So unlike the old EM tables, you *won't* find anything in the Visual Basic script for a ROM-based game for most DOF events. Most DOF events are instead keyed to the things that the ROM controls through VPinMAME: the kicker solenoids, the playfield

lamps, the rollover switches, etc. There's no point in searching through the Visual Basic script for *Medieval Madness* to find out where the script deploys the trolls, because it doesn't; the ROM software deploys the trolls.

If you can't find this stuff in the Visual Basic scripts, where do you look for it, then? Well, we *could* look at the Visual Basic equivalent in these games - namely the ROM scripts themselves. And you can, if you're good at reading 6802 machine language in binary/hex format. But you sure wouldn't want to. And fortunately we don't have to.

The better way to do this is to go to the documentation. The pinball makers were pretty good about documenting everything that made up their machines, and relating it back to the software. So we can use the original pinball machine manuals, in combination with some knowledge about how VPinMAME works, to piece together how the ROMs work without actually having to decode the ROMs.

The rest of this section explains how to find the necessary information in the original pinball machine documentation, and how to relate it to the DOF codes.

Get the Operator's Manual

Before you go on, you should find a copy of the original Operator's Manual for the game you want to customize. Almost every real pinball machine has one, and you can usually find them online. The best place to start looking is IPDB, which has detailed entries for most pinball machines ever made. Find the entry for the game you're looking for, and look to see if there's a link to the manual. If you can't find it there, a simple Web search (e.g. "Lost World pinball operator's manual") will often turn it up.

The Operator's Manual usually has the key information required for DOF customization. The manual generally has lists of all of the switches, lamps, and solenoids, as well as instructions on how to use the game's operator menu to adjust game settings and run diagnostics.

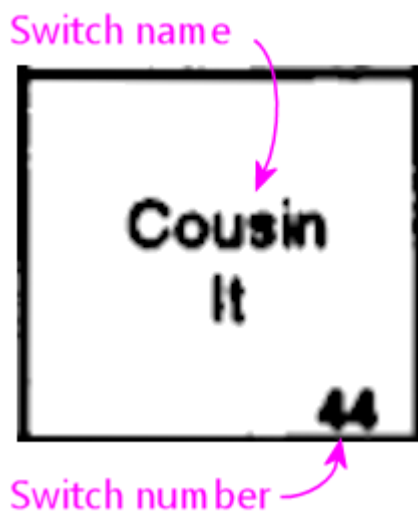
How to identify switch numbers in ROM tables

The DOF trigger code for switches is **W** (because "S" was already taken, for Solenoids), so W17 triggers an effect when switch 17 is hit.

To identify a switch for DOF purposes, you need its switch number in the original ROM software. This is listed in the operator's manual in the **Switch Matrix** table. There's usually a copy of the switch matrix at the very front or very back of the book, and it typically looks something like this:

Switch Matrix Table										
Dedicated Grounded Switches	Column Row	White → ← Green								Flipper Grounded Switches
		1 Green-Brown J206-1 U20-18	2 Green-Red J206-2 U20-17	3 Green-Orange J206-3 U20-16	4 Green-Yellow J206-4 U20-15	5 Green-Black J206-5 U20-14	6 Green-Blue J206-6 U20-13	7 Green-Violet J206-7 U20-12	8 Green-Gray J206-8 U20-11	
Orange-Brown J205-1 Left Coin Chute	01	1 White-Brown J206-1 U18-11	Not Used	Blam Tilt	Upper Left Jail	Grave "G"	Shooter Lane Enter	Swamp Lock Upper	Bookcase Open	Black-Green J205-1 Right Flipper End of Stroke
Orange-Red J205-2 Center Coin Chute	02	2 White-Red J206-2 U18-9	Not Used	Coin Door Closed	Upper Right Jail	Grave "R"	Not Used	Train Whistle	Swamp Lock Center	Blue-Violet J205-1 Right Flipper Button
Orange-Black J205-3 Right Coin Chute	03	3 White-Orange J206-3 U18-5	Start Button	Ticket Opto	Center Left Jail	Chair Kickout	Bookcase Opto 1	Thing Exit Lane	Swamp Lock Lower	Black-Blue J205-2 Left Flipper End of Stroke
Orange-Yellow J205-4 4th Coin Chute	04	4 White-Yellow J206-4 U18-7	Plumb Bob Tilt	Always Closed	Center Left Jail	Cowboy II	Bookcase Opto 2	Right Ramp Enter	Lockup Kickout	Blue-Gray J205-2 Left Flipper Button
Orange-Green J205-5 Normal Function Reverses Credit	05	5 White-Green J206-5 U19-11	Left Trough	Right Flipper Lane	Lower Jail	Lower Swamp Milton	Bookcase Opto 3	Right Ramp Top	Left Outline	Black-Violet J205-4 Upper Right Flipper End of Stroke
Orange-Blue J205-6 Normal Function Yellow Demos	06	6 White-Blue J206-6 U19-9	Center Trough	Right Outline	Left Slingshot	Not Used	Bookcase Opto 4	Left Ramp Top	Left Flipper Lane 2	Black-Yellow J205-3 Upper Right Flipper Button
Orange-Violet J205-7 Normal Function Up	07	7 White-Violet J206-7 U19-5	Right Trough	Ball Shooter	Right Slingshot	Center Swamp Milton	Bumper Lane Opto	Upper Right Loop	Thing Kickout	Black-Gray J205-5 Upper Left Flipper End of Stroke
Orange-Gray J205-8 Normal	08	8 White-Gray J206-8 U19-3	Center Trough	Not Used	Upper Jail	Upper Jail	Right Outline	Left Outline	Left Outline	Black-Blue J205-5

The "matrix" refers to how the switches are physically wired in the original machines, but we don't need to know any of the wiring details for virtual purposes. We just need the software ID for the switch, which should be printed somewhere in the box, usually at bottom right:



That's the switch number used in the original program ROM, which is also the switch ID used in VPinMAME. So we can use these numbers directly in DOF **W** codes. For example, this tells DOF to fire an effect for 500ms when switch 44 is hit:

```
W44 500
```

There are two special columns in the example switch matrix above that we need to mention.

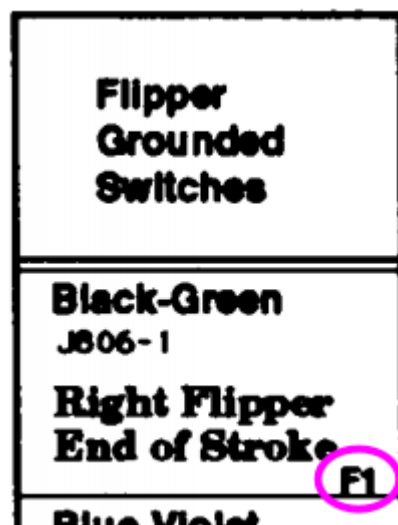
The first special column is the one at the far left labeled "dedicated grounded switches". These were used in almost all of the games in the 1980s and 1990s, so you'll see this first column all the time. The name refers to how they're physically wired in the original game, which doesn't concern us for virtual purposes, but we do still care about the numbering. And the numbering can look weird when they list it in these tables:

Dedicated Grounded Switches	
Orange-Brown (1) J205-1 Left Coin Chute	D1
Orange-Red J205-2	V4

It's a little blurry, but yes, you're reading that right - it says "D1". That doesn't work for DOF, because DOF only allows you to enter a number there. Fortunately, there's a

pretty standard way of dealing with this, because the "dedicated grounded switches" are so common in games from the 1980s and 1990s. VPinMAME reserves switch numbers 1-8 for these special switches. You can simply drop the "D" and you have the DOF number, so D1 is 1 in DOF. Note that the labeling in the operator's manual might be something other than a "D" prefix; whatever it is, it's usually safe to assume that the eight switches in the "dedicated" column are simply numbered 1-8 in VPinMAME and DOF, in the order they're listed from top to bottom.

The second special column is the one at the far right labeled "Flipper grounded switches". You'll generally only see this one on games made in the 1990s; in the 1980s, the the flipper buttons were wired directly to the flipper coils, not to the CPU. The 1990s games controlled the flippers through the CPU, so the flipper buttons and the limit switches on the flippers had CPU connections. That's what you see here. As with the dedicated grounded switches on the left, these buttons often have funny numbering in the tables that doesn't work with DOF, in this case "F" numbers:



As before, we have to translate these to something that DOF can use. VPinMAME handles these switches by assigning them to a numeric range above all of the matrix switches, starting at 111. So the first switch in this column is 111, the second is 112, and so on.

How to identify lamp numbers in ROM tables

The DOF trigger code for lamps is **L**, so L19 triggers an effect when Lamp 19 is lit.

Before we start explaining how to find lamps in general, you should note that *flasher* lamps - the bright lamps enclosed in plastic domes on a playfield - aren't usually "lamps" at all, at least as far as the operator's manuals are concerned. They're usually listed under "solenoids" or "coils". This is a quirk of the electronics used in the 1980s-90s machines. The control circuitry for lamps could only handle low-power bulbs. The bigger, ultra-bright bulbs they used for the flashers were too big for the regular lamp circuits. The circuitry that *could* handle such large loads was the solenoid drivers. So they wired the flashers to the solenoid boards. From the control software's perspective, that made them "coils", so that's how they're listed in the manuals.

For regular lamps (not flashers), find the **Lamp Matrix** table in the operator's manual. There's usually a copy near the very front or very back of the book. It usually looks something like this:

Row	Column	1 Yellow-Brown J137-1 Q98	2 Yellow-Red J137-2 Q97	3 Yellow-Orange J137-3 Q96	4 Yellow-Black J137-4 Q95	5 Yellow-Green J137-5 Q94	6 Yellow-Blue J137-6 Q93	7 Yellow-Violet J137-7 Q92	8 Yellow-Gray J138-9 Q91
1	Red-Brown J133-1 Q90	Thing Multiball	Upper Left Jet	G-R-E-E-D "G"	Not Used	Thing	Left Special	Lite Advance X	"Thing" ""-1
2	Red-Black J133-2 Q89	Extra Ball	Upper Right Jet	G-R-E-E-D "R"	Advance X	Raise The Dead	Lite Thing Flips	Right Special	"Thing" "T"
3	Red-Orange J133-4 Q88	Jackpot	Center Left Jet	G-R-E-E-D "E"-1	Grave "G"	Lite Extra Ball	Lite 2 Bear Kicks	Shoot Again	"Thing" "H"
4	Red-Yellow J133-5 Q87	Grave "A"	Center Right Jet	G-R-E-E-D "E"-2	Grave "R"	House 6 Million	Electric Chair Yellow	Vault Green	"Thing" "I"
5	Red-Green J133-6 Q86	Stars	Lower Jet	G-R-E-E-D "D"	The Mammoth	Quick Multiball	House "?"	Vault Red	"Thing" "N"
6	Red-Blue J133-7 Q85	Super Jackpot	Cousin It	5X Graveyard	Swamp Lock	Feeler's Tunnel Hunt	House 9 Million	Not Used	"Thing" "Q"
7	Red-Violet J133-8 Q84	Grave "V"	2 Bear Kicks	Center Swamp Million	Electric Chair Red	House Seance	Graveyard At Max	Thing Yellow	"Thing" ""-2
8	Red-Gray J133-9 Q83	Upper Swamp Million	Thing Flips	Lower Swamp Million	Grave "E"	Hit Cousin It	House 3 Million	Thing Green	Credit Button

The row-and-column format refers to how the lamps are wired physically in the original games - remember that the operator's manual is primarily for people maintaining and repairing the machine. For virtual purposes, we can remain blissfully ignorant of the color stripes on the wires connecting to the lamps and the orientation of the matrix diodes. But there's still one piece of information in this table that's useful to us: the lamp number.

If you look at the bottom right corner of each box, there's a little number printed:



That's the lamp number, which is used to identify the lamp in the ROM software. VPInMAME thankfully uses the same numbering scheme that the ROM software uses, so you can use these lamp numbers directly in DOF using the **L** code. This tells DOF to fire an effect for 500ms when lamp 72 is hit:

```
L72 500
```

How to identify solenoid numbers in ROM tables

The DOF trigger code for solenoids is **S**, so S19 triggers an effect when solenoid 19 fires.

Solenoids are trickier to identify than switches and lamps, because the numbers in the Operator's Manual don't usually match the numbers in VPinMAME. VPM assigns its own numbering to the solenoids instead. DOF uses the VPM numbering, not the original Operator's Manual numbering, so in order to set up a DOF config, you need to figure out the VPM numbers.

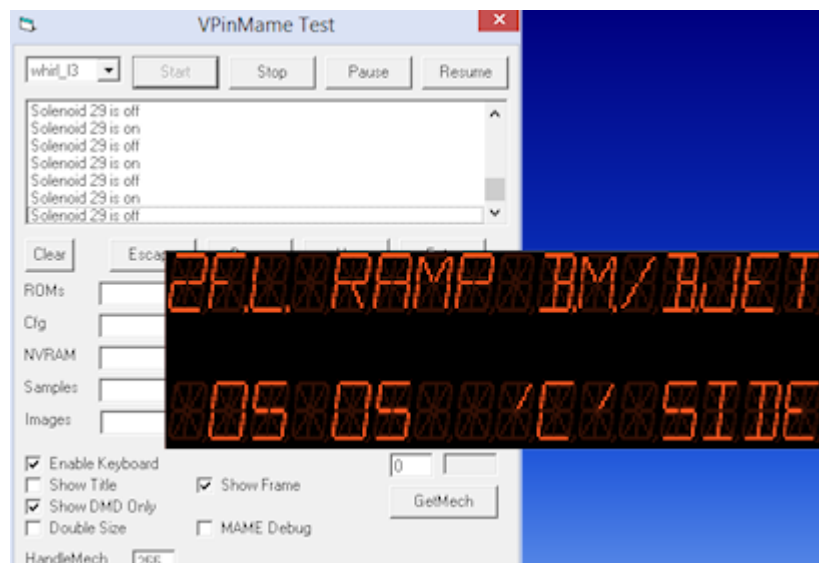
For example, the Operator's Manual might call the outhole kicker Solenoid "2C", but VPM (and thus DOF) might call it Solenoid 32.

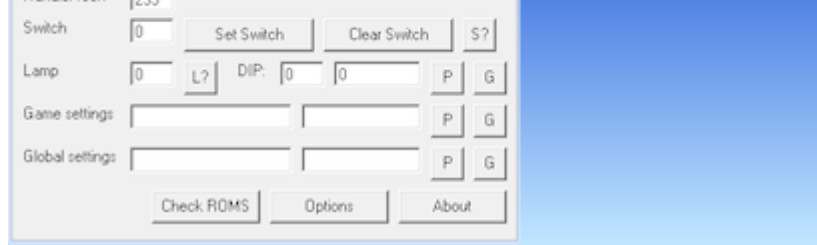
It would be a lot nicer if VPM could use the original numbering, but the designers of the original tables in the 1980s and onward made that too difficult, by treating the solenoid numbers as somewhat arbitrary labels. So we're stuck with this extra work of figuring out the VPM numbering for each table.

The easiest way to identify the solenoids is to actually run the game's ROM, but **not** using Visual Pinball. Instead, we're going to use a special test program, called VPinMameTest. This should be located in the VPinMAME folder under your Visual Pinball program folder.

Here's the basic approach we're going to use. VPinMameTest has a little window that shows messages like "Solenoid 1 is on" or "Solenoid 7 is off" whenever a solenoid changes state. The solenoid numbers displayed there are the PinMAME solenoid numbers, and thus the DOF "S" numbers. If the window says "Solenoid 19 is on", we know we're talking about DOF "S19". So what we're going to do is load the table that you want to map into VPinMameTest and then fire each of its solenoids in turn.

But how do we get the solenoids to fire? We're going to use the game's own built-in test menu. Almost every electronic pinball from the mid 1980s onward has a set of operator menus for adjusting game settings and running diagnostics. The diagnostics usually include a solenoid test mode, which lets you cycle through all of the solenoids in the game and fire each one. On the real machines, repair people would use this to test the mechanical action on the playfield to make sure that each coil is firing like it should be. We're going to do the almost same thing, but instead of watching real coils fire, we're going to watch VPinMameTest fire its virtual coils. The test menu will show us a message on the pinball DMD saying which solenoid it's firing - it'll say something like OUTHOLE KICKR 05. VPinMameTest will simultaneously show "Solenoid 19 is on, solenoid 19 is off," etc. This reveals the association between the ROM solenoids and the MAME numbers: we now know that OUTHOLE KICKR 05 is MAME solenoid 19, and thus DOF S19.





So let's get started. The first thing to do is fire up VPinMameTest.

(Note: If you get an error when you run this program saying "MSSTDFMT.DLL is missing", you'll need to download that file and copy it to the same folder containing the VPinMameTest program. You can find a copy of it [here](#).)

Once VPinMameTest is running, go to the drop list at the top left corner. Select the ROM for the table you want to map. The ROM name is usually an abbreviation of the table name. It might take a little guesswork to figure out which is which.

You'll need to have the selected ROM file installed in your VPinMAME/ROMs folder in order to run it. The drop list somewhat confusingly shows every ROM that VPinMAME has ever heard of, whether you have any of them installed or not. But you can only run the ones that are actually present on your machine.

Once you select the desired ROM, click Start. This will fire up the ROM, and the appropriate pinball display should appear on your screen. This will be the usual 7-segment LED display, alphanumeric LED, or DMD style, depending on the game. At this point, most games will run through their power-on self test and go into attract mode. Some games might report errors, such as switch problems or "Missing Pinball". This is simply because the ROM is running without a physical pinball machine attached - it's like you took the CPU board out of the pinball machine and ran it without any of the cabinet wiring attached. But this is okay for what we're doing here.

The next step is to enter the ROM's test menu. This is where things get a little tricky, because every game has its own way of doing this. Fortunately, there are commonalities among machines within each generation, so we can offer some general instructions that will work on most machines. Scan down the sections below to find the type of machine you're working with.

Williams WPC games (1990s-2000s)

These games have a fairly friendly menu interface. The first thing you'll need to do is press End, to simulate opening the coin door. This will give you access to the operator buttons:

- 7 = Cancel
- 8 = Previous/-
- 9 = Next/+
- 0 = Enter

Press 0 (Enter). This will display the game name for a few moments, then prompt you to press 0 (Enter) again to access the menu. Press 9 (Next) until you see "Tests" displayed. Press 0 (Enter). Press 9 (Next) until you see "Solenoid Test", then press 0 (Enter).

At this point, the menu will show you the first solenoid, and will fire it every couple of seconds. The pinball display will show something like "AUTO PLUNGER, T.4 01 REPEAT". AUTO PLUNGER is the name of the selected solenoid, T.4 is just a label for

the current menu, and 01 is the ROM solenoid number, which will also be the number used in the Operator's Manual. If you go over to the VPINMameTest window, you should see "Solenoid 1 is on, Solenoid 1 is off" repeating every couple of seconds. The game is firing the solenoid selected in the menu, so this tells you that ROM solenoid 01 is the same as MAME solenoid 1 = DOF "S1".

Press 9 (Next) to move to the next solenoid. Write down each mapping and repeat until you've visited all of the solenoids.

Williams System 11 (late 1980s to early 1990s)

Be warned: the operator interface on these machines isn't very friendly. There's a logic to it, but it's a bizarre and twisted logic, driven by an evil assembly programmer's whims, rather than anything that makes sense to a user. I've owned a couple of real System 11 machines for nearly 20 years and I still have fits finding my way around their stupid menu systems.

The operator controls on these machines consist of two buttons: "AUTO/UP - MANUAL/DOWN" (7 in MAME) and "ADVANCE" (8 in MAME). The START button (1 in MAME) on the front of the machine also gets involved in places. (There's a third button, as you can see in the photo, but its only function is to reset the high scores. Why they dedicated a whole button to this, rather than using it to make menu navigation easier, supports my belief that the programmer had evil intent.)



AUTO/UP - MANUAL/DOWN is what makes the system so confusing. There are two things you have to know about it. The first is that it's a **toggle** button. On the real machines, it works physically just like the clicker on a ball-point pen, in that it cycles between the "up" and "down" positions each time you push it. So you can tell by looking at it which position you're in. In MAME, it still has this toggle effect each time you push the "7" key, but of course it's just a keyboard key, so you can't tell by looking at it which mode it's in. You just have to remember. Good luck!

The second thing you have to know about AUTO/UP - MANUAL/DOWN is that it means three different things, depending on context. What they mean by the name is this: sometimes it's the UP/DOWN button, and sometimes it's the AUTO/MANUAL button. And they didn't even have room to print this on the label, but sometimes it's the ADJUST/TEST button. Got it? Probably not... so here are the contexts where the modes apply:

- When you're **not in any menu** (in other words, when the game is in attract mode), it's the TEST/ADJUST button. When it's in the "up" position, pressing the ADVANCE ("8") button from attract mode takes you into the SETTINGS menu. When it's in the "down" position, pressing ADVANCE ("8") from attract mode takes you into the TEST menu.
- When you're in the **settings menus**, it's the UP/DOWN button. It controls whether the ADVANCE button moves forward or backwards through the menu system.
- When you're in the **test menus**, it's the MANUAL/AUTO button. In AUTO/UP mode, each test cycles through all of its different items automatically, advancing to the next item every 2 seconds or so. In MANUAL/DOWN mode, no automatic cycling takes place. Instead, the system stays on the same test item until you press ADVANCE. (And note that ADVANCE moves to the NEXT test item in this case, even though it seems like we should be in DOWN mode.) moves to the next menu item, and DOWN mode makes ADVANCE move to the

previous menu item.

So now that you know how this crazy UI works, let's outline the strategy for mapping out the solenoids. We're going to enter test mode (MANUAL/DOWN + ADVANCE), go forward through the test menus until we get to the solenoids section (AUTO/UP + ADVANCE until we get to solenoids), then manually step through each solenoid in the system. For each, we'll observe the ROM name on the alphanumeric display and note the MAME solenoid number displayed in the test window. We'll make a note of each, then press ADVANCE to move on to the next. Repeat until we know the MAME number for each ROM solenoid.

- Press 7 (DOWN mode)
- Press 8 (ADVANCE) - this should enter the test menu
- Press 7 (UP mode)
- Press 8 (ADVANCE) until you see COIL TEST
- Press 7 (MANUAL mode)

It's sometimes hard to get the mode right initially, so you might find yourself in the audit/setup menu rather than the test menu. If so, you can get back to attract mode by pressing 8 repeatedly until you go past the last adjustment item. If you're stuck in a loop going backwards (the menu item number keeps decreasing), press 7 to switch directions and try 8 again.

You should now see something like this on the alphanumeric display: OUTHOLE 05 01 'A' SIDE. This is telling you that the current solenoid being tested is the outhole kicker, labeled as solenoid 01A in the game's Operator's Manual. You might want to look at the solenoid table in the manual at this point to verify that it matches.

The VPINmameTest window should be displaying "Solenoid 1 is on, Solenoid 1 is off," repeating every couple seconds. This is because the ROM is firing the currently selected solenoid repeatedly. This tells us that the OUTHOLE Solenoid 01A is VPINMAME solenoid 1 and DOF "S1". Write down the association.

When you've noted this solenoid, press ADVANCE to move to the next one. The alphanumeric display will update to the next solenoid name and number, and the VPINmameTest window should start displaying a new solenoid number switching on and off. Write this one down.

Repeat until you have all of the numbers mapped out.

Williams System 9 (mid 1980s)

The System 9 games are very similar to the System 11 games in the menu design, with the big difference that they lack the alphanumeric display. These games have simple 7-segment numeric displays only, so their ability to display context information is severely limited. But the structure of the menus is nonetheless similar.

You should be able to activate test mode with the same sequence of keys as in the System 11 machines, outlined above. You'll know you're in test mode when you start seeing all of the numeric displays cycle through a sequence like this: 0000000, 1111111, 2222222, 3333333... That's the display test. Press 8 to advance to the next test. The exact location of the coil test can vary, but on many it's the third test, so you'd have to press 8 two more times. You'll know you're at the coil test when you start seeing "Solenoid X is on, Solenoid X is off" messages in the VPINmameTest window.

At this point, press 7 to switch to MANUAL mode. The display will show something like "02 10". The "02" is the menu position number, telling you you're in coil test

mode; it might be some other number, but it'll stay the same throughout the coil test mode. The "10" is the solenoid number currently being tested. As with the System 11 games, you can write down the association between the number shown here and the "Solenoid X is on" number in the test window. Once you've noted the mapping, press 8 to advance to the next solenoid. When you reach the last solenoid, the menu will loop around to solenoid 01 and start over.

Williams System 3-8 games (1970s to mid 1980s)

As you might expect, the operator controls on these early electronic games are even more primitive and peculiar than the System 9 games, but a lot of the basic structure is still similar. The big difference is that there's another key, the **DIAGNOSTICS** button, which VPinMAME maps to the 9 key. Press this button and you'll enter test mode. As before, you can probably find your way to the solenoid test with a little trial and error. The 7 and 8 keys generally work just like in System 11 games, as the **AUTO/UP-MANUAL/DOWN** and **ADVANCE** buttons, respectively.

Everything else

There are too many different systems to include every possible one here. If the type of game you're trying to map isn't covered in the sections above, the best advice I can offer is to find the table's Operator's Manual online and look for instructions on running diagnostics. (If anyone wants to send me instructions for a particular manufacturer/generation that they know all about, I'd be happy to add them to this section.)

The caveat with that advice is that you'll have to figure out how the buttons mentioned in the manual map to MAME keys for yourself, probably by trial and error. The manual will undoubtedly point you to some special-purpose buttons, with names like **TEST**, **DIAGNOSTICS**, **ADVANCE**, **SELECT**. MAME should have mappings for those keys, but I don't think there's a list anywhere of what these mappings are for all games, and my impression is that the mappings are all ad hoc, without any real master plan. The one ray of hope here is that operator keys are almost always mapped to 7, 8, 9, and 0, so the trial-and-error search space is relatively small. Try pressing those buttons to see what reaction you get. The Operator's Manual will describe what you should see on the display (if anything) in the various menu modes, so try the 7-8-9-0 keys until you get the expected response.

If you do happen to figure out the procedure for a group of machines not covered above, I'd be very happy to add your findings to the collection here, so please send them my way.

Appendix 8. Coin Door Interface Board

In the Chapter 40, Coin Door chapter, we discuss the special wiring plug that the Williams WPC coin door uses. That plug is wired to all of the switches and lamps in the coin door: the coin chute switches that detect when you've inserted a quarter, the slam tilt switch, the service control panel buttons, and the coin slot lamps.

In a virtual cab, we have to connect that special coin door plug to the key encoder, so that the buttons and switches in the coin door can be used as inputs to the virtual pinball software. We also have to connect it to a power supply, to light up the coin slot lamps.

In the original pinball machines, the coin door connector plugged into a little circuit board located near the front of the cabinet, whose job was simply to interface the coin door plug to the rest of the wiring in the machine. The tidiest approach in a virtual cab is to do the same thing, by installing a little interface board of our own that plugs into the coin door connector on one end, and connects to your key encoder on the other end.

I drew up plans for a circuit board to accomplish this. It's a simple design that you build yourself. This section has download links, parts lists, and instructions for assembling and installing it.

6.3V Power Supply

The Williams coin doors usually come equipped with #555 incandescent bulbs installed in the coin slots, to light up the slots. Those bulbs require an unusual voltage level, 6.3V. You probably don't have a 6.3V power supply in your pin cab, unless you installed one specifically for these sorts of lamps.

To help with this, I included a 6.3V power regulator in the circuit board design. If you build the full board design, you'll get the 6.3V supply automatically; you just have to plug in power from a 12V supply (which you probably do already have, since it's one of the voltage levels that you get from an ordinary ATX PC power supply).

However, the 6.3V regulator is optional. You can simply omit all of the parts for it (leaving their slots on the circuit board empty). If you do this, you can still get the lamps to light up by using one of these options:

- Power the lamps with 5V. #555 incandescent bulbs will work on 5V; they just won't be as bright as they're meant to be. If you don't mind that they're not as bright, this is easy, and you can skip buying a few parts.
- Use 5V as above, but also replace the incandescent #555 bulbs with LED equivalents. You can buy plug-compatible #555 LED bulbs, and most of those will work on 5V (even though they'll nominally be 6.3V bulbs, since they're meant to be plug-in replacements for the incandescent bulbs). LEDs don't usually show as much drop in brightness as incandescents at lower voltages; if they light up at all, they should appear to be pretty much normal brightness.
- Power the lamps with a separate 6.3V power supply. If you already have a 6.3V supply, you can plug it in to the board and the board will pass it through to the lamps. One easy way to set up a 6.3V power supply is to buy an adjustable DC-to-DC step-down regulator board on eBay, and set it to 6.3V. If you're not fond of soldering, you might prefer this to soldering the extra parts for the on-board 6.3V regulator on the coin door interface board.

Version 2

This is a new version of the board that I drew up in February, 2021. I had to come up with this new design because the 6.3V regulator chip used in the original version 1 design is no longer in production. The old design is still available, of course; see Version 1 below. Feel free to use the old design if you prefer it, but just be aware that you might not be able to find the 6.3V regulator chip it calls for.

Warning: This new version hasn't been tested yet. If you'd like to try building one and let me know how it works, that would be very helpful! Just be aware that it might have design flaws that I haven't caught yet. If you're not feeling so adventurous, you might want to stick with the version 1 design.

EAGLE plans: mjrnet.org/pinscape/downloads/WilliamsCoinDoorConnector-v2.zip

Parts list: See Chapter 91, Electronic Parts List.

Screw terminals or pin headers: The parts list calls for Phoenix Contact screw terminals for JP1 and JP2, which are the places where you connect the wiring that goes out to your key encoder and power supply. I used the screw terminals in the design because many pin cab builders like the simplicity of wiring them. With screw terminals, you just strip a quarter inch or so from the end of the wire, insert the end of the wire into the terminal, and tighten the screw to clamp it in place. No special tools are required.

The parts list links to Phoenix Contact parts on Mouser.com for the screw terminals, but you can also find cheaper generic versions here: www.pololu.com/category/177/0.1-2.54-mm-screw-terminal-blocks.

If you prefer, you can use standard 0.1" straight pin headers in place of the screw terminals. I personally prefer pin headers in most cases, because they mate with pluggable connectors that can be easily plugged and unplugged. With screw terminals, you have to attach each wire separately, which is time-consuming if you have to remove the whole board for any reason. The Phoenix Contact terminals and the standard 0.1" pin headers will both fit physically, so you can use whichever type of connector you prefer. See Chapter 81, 0.1" Pin Headers for more about those connectors.

If you do opt for 0.1" pin headers, be aware that JP1 can accommodate a 10-pin header, even though the parts list calls for a 9-position screw terminal. I intentionally added drilling for an extra "dummy" pin, so that you can use a 10-pin header. The extra 10th pin isn't connected to anything electrically on the board; it's just there so that a 10-pin header will fit. The reason you might want to use a 10-pin header instead of a 9-pin is that it's hard to find the mating "crimp pin housing" in a 9-pin size, whereas it's easy to find 10-pin housings.

To manufacture the board: simply upload the .brd file from the plans above to OSH Park, or a different PCB maker or your choice. (Other PCB makers usually require Gerber files rather than .brd files. You can create those with the free version of EAGLE.)

To build the board *with* the 6.3V regulator: Solder all of the listed parts to the board.

To build the board *without* the 6.3V regulator: Don't install any of the voltage regulator parts (C1, C2, IC1, R1, R2). Install only the pin headers and screw terminals.

Connections: Once you've built the board, mount it near the coin the door where

you can plug in the connector from the door. Connect power for the coin chute lamps as described above (use a 12V power supply input if you included the voltage regulator parts, or a 5V or 6.3V supply if you didn't). Plug the 13-pin connector from the coin door into the mating pin header on the board.

The terminals on JP1 all connect to your key encoder, to allow the switches in the coin door to send button input signals to the virtual pinball software on the PC. The switch outputs on the board are labeled as follows:

- L = left coin switch
- M = middle coin switch (not used on US two-chute coin doors)
- R = right coin switch
- Cn = Service Cancel/Escape
- - = Service Down/-
- + = Service Up/+
- En = Service Enter/Select
- T = Slam tilt switch
- Co = all-switch common wire

The terminals on JP2 connect to power for the coin slot lamps. There are several ways to connect these terminals, depending on (a) whether or not you installed the 6.3V regulator parts, and (b) whether you want the coin slot lamps to be on all the time or controlled by software on the PC (via Chapter 46, DOF). Here are the instructions for each combination:

- **With 6.3V regulator, lamps always ON:**

- +LAMP not connected, **or** you can use +LAMP as a 6.3V power source for up to four additional #555 bulbs
- LAMP- to 12V power supply (-) terminal (0V/GND, black wire on ATX disk plugs)
- -12V same as LAMP-
- 12V+ to 12V power supply (+) terminal (yellow wire on ATX disk plugs)

- **With 6.3V regulator, lamps controlled by DOF:**

- +LAMP not connected, **or** you can use +LAMP as a 6.3V power source for up to four additional #555 bulbs
- LAMP- to feedback controller output port (LedWiz, Pinscape, etc) assigned to coin slot lamps
- -12V same as LAMP-
- 12V+ to 12V power supply (+) terminal (yellow wire on ATX disk plugs)

- **No 6.3V regulator, lamps always ON:**

- +LAMP to your 5V or 6.3V power supply (+) terminal (red wire on ATX disk plugs)
- LAMP- to your 5V or 6.3V power supply (-) terminal (0V/GND, black wire on ATX disk plugs)
- -12V not connected
- 12V+ not connected

- **No 6.3V regulator, lamps controlled by DOF:**

- +LAMP to your 5V or 6.3V power supply (+) terminal (red wire on ATX disk plugs)
- LAMP- to feedback controller output port (LedWiz, Pinscape, etc) assigned to coin slot lamps
- -12V not connected
- 12V+ not connected

If you didn't build the 6.3V regulator, you can use either a 5V or 6.3V power supply to power the coin slot lamps. 6.3V is preferable since that's the voltage the bulbs are designed for; they'll work at 5V but won't be as bright. LED replacement bulbs might let you use 5V without loss of brightness, so you might want to try that if you don't want to bother with adding a separate 6.3V supply.

If you're using DOF to control the lamps, make sure that the output controller port that you're using to control the lamps has enough power load capacity. Two incandescent #555 bulbs will consume about 500mA. That's safe for any Pinscape flasher port or power board port, and it's right at the limit for an LedWiz port. (So it might work with an LedWiz, but you'll be pushing your luck a bit; a booster circuit in this case would be a good idea. See the Chapter 50, LedWiz chapter for help on adding booster circuits.) If you replaced the bulbs with LEDs, they'll use much less power, probably about 30mA per bulb at most, which makes them safe to use with an LedWiz directly, with no booster.

Version 1

This section covers my original version of the board. If you plan to use the on-board 6.3V power supply feature, you might want to consider building the new design above instead, because the voltage regulator chip that this design uses is out of production and hard to find. The new design uses a newer chip that's still available.

EAGLE plans: mjrnet.org/pinscape/downloads/WilliamsCoinDoorConnector.zip

Parts list: See Chapter 91, Electronic Parts List.

If you don't want to include the on-board 6.3V power supply feature, you can omit the parts C1, C11, Q1, and JP11. Simply leave the slots for those parts on the board empty. If you already have a separate 6.3V supply, you can omit all of these and instead supply the board with 6.3V from the external supply.

Warning: the regulator chip Q1 is no longer in production, and the electronics vendors no longer sell it. It might still be possible to find surplus parts on eBay, although that can be spotty in terms of quality. Unfortunately, I don't know of any suitable substitute for the chip that will work with this board design. That's the whole reason I designed the new version 2 board above; there *are* other chips that will do the job, but they all require slightly different circuit designs, so I had to change the board layout a bit.

To manufacturer the board: simply upload the .brd file from the plans above to OSH Park, or a different PCB maker or your choice. (Other PCB makers usually require Gerber files rather than .brd files. You can create those with the free version of EAGLE.)

To build the board *with* the 6.3V regulator: Solder all of the listed parts to the board. Connect 12V from your secondary ATX power supply (the one you use for feedback devices: see Chapter 45, Power Supplies for Feedback) to the two-pin header labeled "12V IN". You can use the two-pin header marked **6.3V** as a power supply for other 6.3V button lamps, as long as you don't exceed 1.5A, or a total of 6

incandescent #555 bulbs (including the ones in the coin chutes).

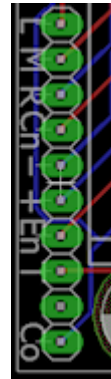
To build it *without* the 6.3V regulator: Don't install any of the voltage regulator parts (C1, C11, Q1, or the 12V input pin header). Install only the main 13-pin header, the 10-pin header for the switch outputs, and the 2-pin header marked **6.3V**. Connect the 6.3V header on the board to your external 6.3V power supply; this will feed the 6.3V directly to the coin chute lamps.

Once you've built the board, mount it near the coin the door where you can plug in the connector from the door. Plug in the power connection for the coin chute lamps (the 12V input if you included the voltage regulator, or the 6.3V input if you didn't). Plug in the 13-pin connector from the coin door.

To wire the switches to your key encoder, you can either solder wires directly to the pin holes for the switch outputs, or (better) you can install a 10-pin header and connect a mating 1x10-pin plug. See Chapter 80, Connectors for details on building the plug using a 0.1" crimp pin 1x10 position housing.

The switch outputs on the board are labeled as follows:

- L = left coin switch
- M = middle coin switch (not used on US two-chute coin doors)
- R = right coin switch
- Cn = Service Cancel/Escape
- - = Service Down/-
- + = Service Up/+
- En = Service Enter/Select
- T = Slam tilt switch
- Co = all-switch common wire



Note that the pin between "T" and "Co" isn't connected to anything on the board. 9-pin housings aren't readily available, so I added an extra unused pin to make it easier to find the matching connector.

Appendix 9. Plywood Cutting Plans for Cabinet Construction

If you're building a pinball cabinet from scratch, the first step is to buy some plywood sheets and cut them up to form the panels making up the cabinet. This chapter provides a few options for how to divide up the plywood.

The standard pin cab plan calls for two thicknesses of plywood: 1/2" for the main cabinet floor and the back of the backbox, and 3/4" for everything else. Most commercial pinball machines use particle board rather than plywood for the 1/2" parts, to cut costs, but it's a nice upgrade to use plywood instead if your budget permits it. In any case, it makes no difference to any of the cutting or building plans.

Index to the layouts:

- Layout 1: Best grouping of same-size pieces
- Layout 2: Single 3/4" sheet plan
- Layout 3: Car-friendly plan

Adjustments for joinery

The cutting plans all assume that you're using mitered corner joins for the main cabinet, where the outer edges of the corners meet at 45° angles. This assumption is important because it means that the finished size of the cabinet on a given side will exactly match the size of the panel on that side, since the panel extends all the way to each corner. If you're using a join that doesn't have mitered corners, such as a rabbet or butt join, each corner will have one piece that doesn't extend all the way to the outside corner, meaning that it will have to be cut a little shorter than the finished cabinet size on that side, to make up for the overlapping section in the join. See Chapter 21, Cab Body - Joinery.

Adjustments for cabinet width

The dimensions shown are all for the standard-width WPC cabinet. If you're building a widebody or custom-width cabinet, you'll have to adjust the widths accordingly.

How to use the plans

Don't start by marking all of the cut lines shown on the plywood sheet. That won't produce accurate results, because your saw blade will remove some of the material between the pieces at each cut, which you can't easily account for in the initial marking. Instead, measure and cut one section at a time. This lets you make the next measurement based on the actual new edge left over from the previous cut.

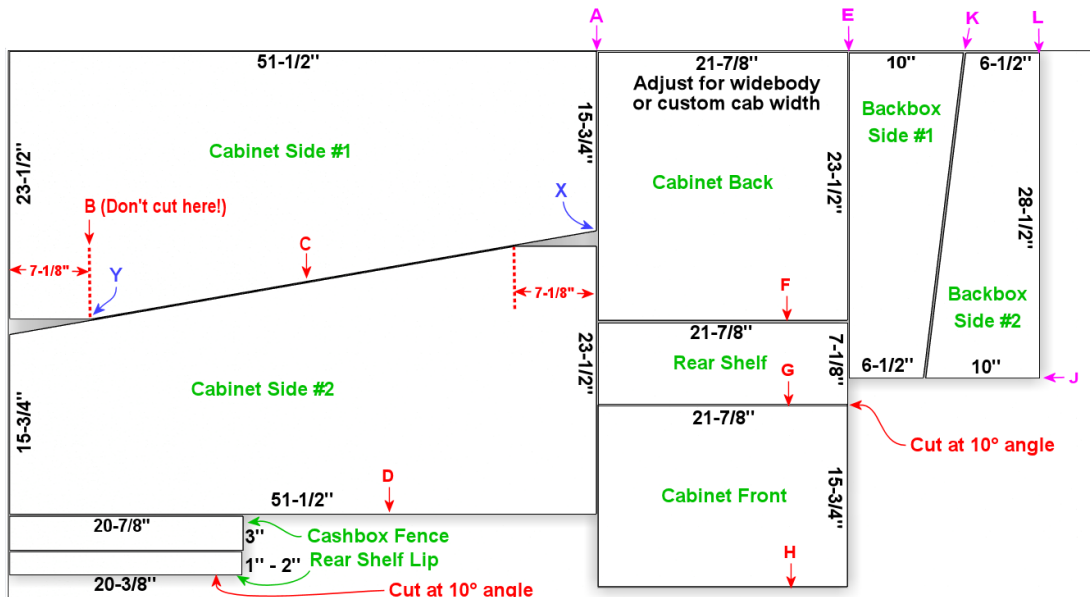
Also, to make sure that your saw blade's width is accounted for in each cut, always align the saw on the "outside" of each cut, so that the blade's width is positioned in the "leftover" portion of the plywood rather than within the work piece you're cutting. Align the edge of the blade on each cut with the outside edge of the work piece.

Layout 1: Best grouping of same-size pieces

This layout groups the pieces in such a way that pieces of the same size are grouped together, so that you only have to make one cut at a given measurement point. The benefit is that it almost guarantees that the grouped pieces will exactly match on

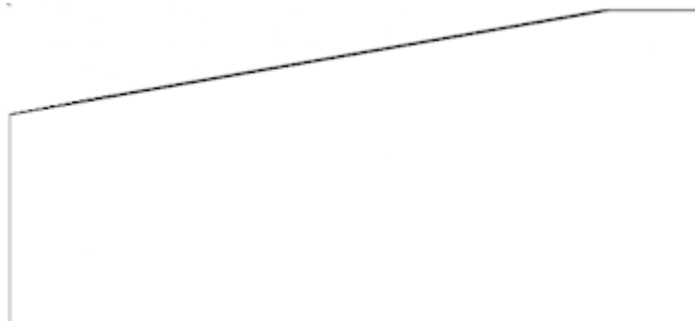
that dimension, since they're all cut along a single line. It ensures that the side walls will end up being exactly the same length front-to-back, the front and back walls are exactly the same width, and the backbox sides are the same height. Another benefit is that this layout is efficient in terms of your time, since it minimizes the number of measurements and cuts.

The downside of this plan is that it doesn't make efficient use of the plywood. It requires two sheets of 3/4" plywood, but it only uses a small portion of the second sheet. You might check with your lumber supplier to see if they sell half sheets or smaller project boards, since you only need about 28" x 18" for the second piece.



Plywood sheet #1 - 4' x 8' x 3/4" nominal thickness. Dimensions are all for the WPC standard-body design.

1. Measure from one edge of the sheet to 51-1/2", to the cut line marked "A" in the diagram above. Cut across the whole sheet on this line. Set aside the smaller portion.
2. We're now going to cut the diagonal line "C" for the side panels. On the "A" cut line, measure 15-3/4" to the point labeled "X", and mark that spot. On the opposite side, pencil in the line marked "B", 7-1/8" from the edge. ("B" is just a reference point - don't cut along this line.) Then measure 23-1/2" along "B" from the edge to the point labeled "Y", and mark that spot. Draw the diagonal line "C" through points "X" and "Y". Cut all the way along "C".
3. The piece you just cut out is one side wall of the cabinet. You just need to cut out that 7-1/8"-long flat portion where the diagonal edge flattens out. Mark a line 23-1/2" from the opposite side, parallel with the opposite side, and cut along that line to make the flat portion. The finished side piece should look like this:

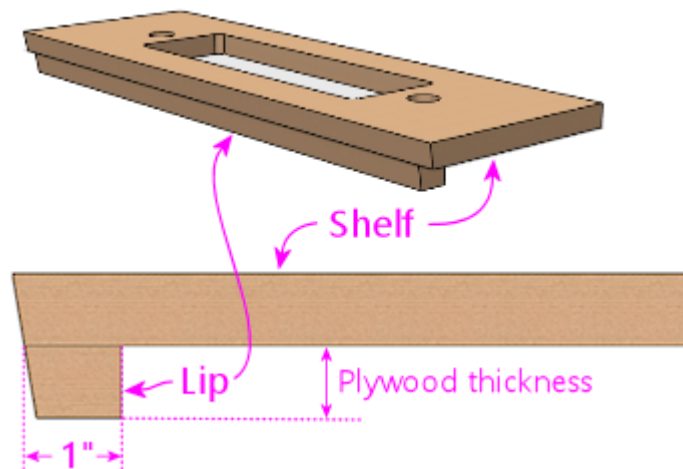


4. On the remaining piece, you can repeat the measurements from step 2 to mark the 51-1/2" line "D", or you can simply use the finished first side piece as a template, since both sides should be identical. Either way, mark line "D" and cut along the line to cut out the second side piece.

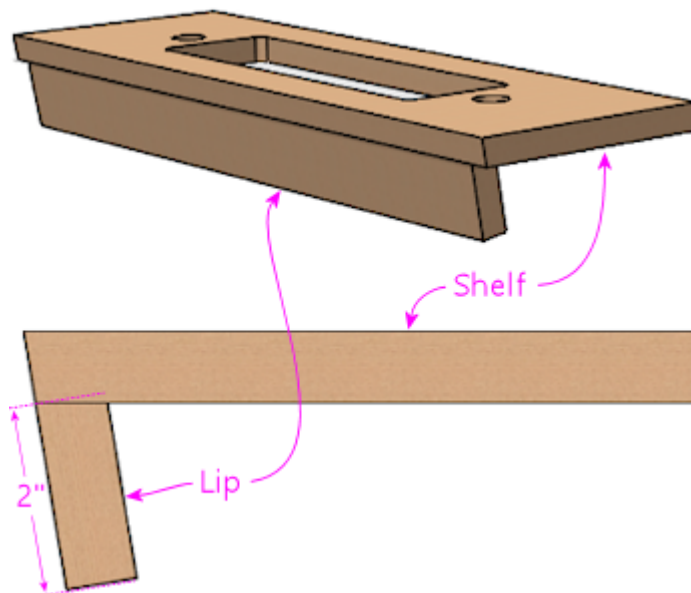
5. Repeat step 3 with the second side piece to cut the 7-1/8" flat portion.

6. From the remains of the 51-1/2"-wide section, cut the two small rectangular pieces shown at the bottom: the cashbox fence (20-7/8" x 3"), and the rear shelf lip (20-3/8" x 1" to 2").

For the rear shelf lip, cut one side with the saw blade at a 10° tilt (bevel angle). This will match the tilt in the front edge of the shelf. I'm giving you a size range for this piece because you'll probably want to adjust it according to your TV setup plans. In the original WPC machines, the lip is set up like this:



For a virtual cab, though, you might want to extend the lip to about 2", by cutting the piece to 2" wide and turning it sideways:



This is just a suggestion - you can make the lip longer or shorter than 2" as needed. The right length depends upon how your TV and other devices (such as a flasher panel at the back) are arranged. The smaller size used on the WPC machines is enough to allow installation of the plastic trim piece that holds the back of the glass cover. On a virtual cab, a longer lip can be used to fill the space between the top of the TV and the top of the cabinet.

7. Now we'll work on the (almost) half-sheet that was left over from the first step, after cutting along line "A". This section should be about 48" x 44". Mark cut line "E" by measuring 21-7/8" from one edge.

For widebody or custom cabinets, adjust this dimension! 21-7/8" is for the WPC Standard Body design, which fits the standard Williams lockbar (Williams/Bally part number D-12615, A-18240). If you're building a widebody that matches the original Williams "Superpin" cabinet, to fit the off-the-shelf WPC widebody lockbar (A-16055, A-17996), the width should be 24-5/8". If you're building to a custom size, you can determine the width to use here by adding 1-3/8" to your desired inside cabinet width (the distance between the inside faces of the side walls - essentially, the width available for the main TV).

Make sure the board is oriented so that "E" goes down the longer (48") dimension! Otherwise you won't have enough material for the three pieces shown.

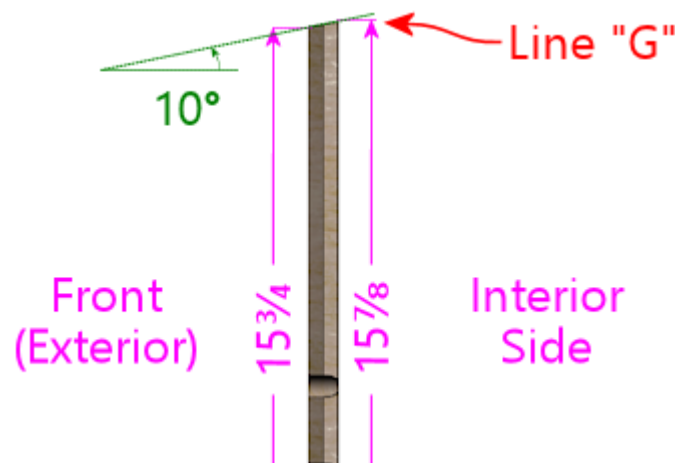
8. Using the 21-7/8"-wide section we just cut, mark line "F" 23-1/2" in from one edge. Cut along this line. This yields the back wall of the cabinet.

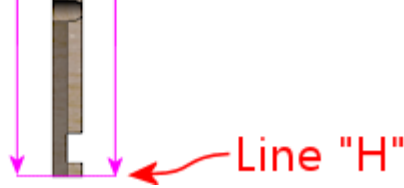
9. Continuing with the remainder of the 21-7/8"-wide section, mark line "G" 7-1/8" from one edge. **Cut along the line with the saw blade at a 10° tilt.** The 7-1/8" line is the **longer** face for the angled cut - the other face should end up about 7" long. This yields the rear shelf for the cabinet.

If your saw can't easily cut at an angle like this, you can get away with making a square cut. Cut the piece at the shorter 15-3/4" size in this case. The angle is to make the top edge of the piece align with the diagonal slant of the cabinet sides, so that all the edges are flush. But this isn't actually all that critical; it's only a slight aesthetic imperfection if you can't make the edges flush, and it's an imperfection you'll almost never have to look at anyway, since the whole area is covered by the lockbar! The only time you'll see it is when you take the lockbar off to get inside for service.

10. Set the saw blade back to straight up (no tilt/bevel angle). Still using the remainder of the 21-7/8"-wide section, mark line "H" 15-3/4" from one edge, using the **shorter face** after taking into account the tilted cut on line "G". Remember that we just cut the one edge of this piece ("G") with a 10° bevel angle. That will be the **top** of the front wall, which needs the angled top edge to match the slope of the side walls. The other side, line "H", which we're cutting now, is the bottom of the front wall.

When marking the cut line "H", be very careful that you're measuring 15-3/4" from the **shorter** side of the beveled edge "G" from the previous step. The distance to "H" on the longer face should end up being about 15-7/8". Here's the side profile of the finished piece, to give you a better idea of what we're going for:





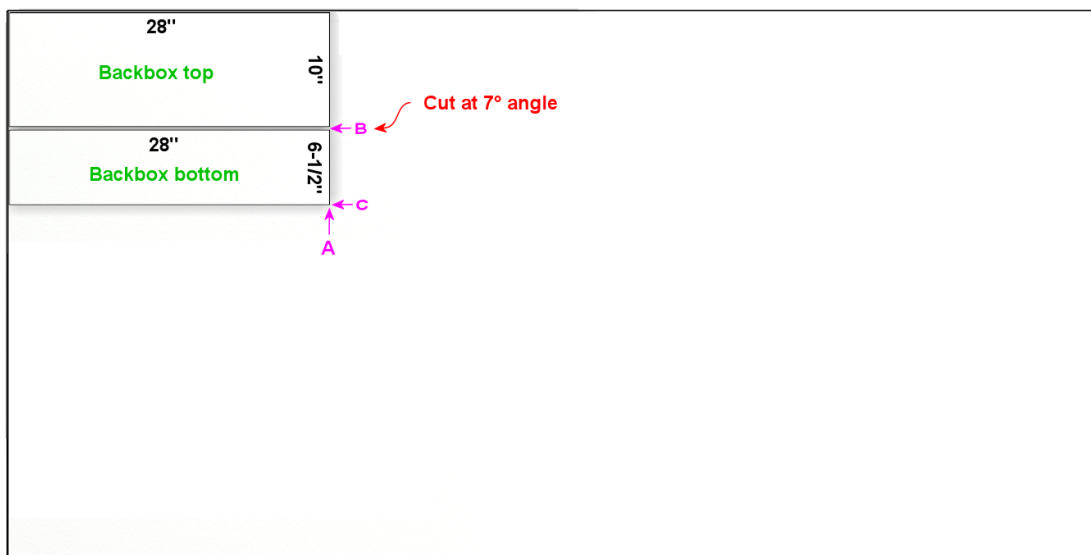
This will use almost the entire rest of the piece, but there should be just enough left that you still have to trim this little bit.

11. There are only two pieces left to cut out of the remainder of the first plywood sheet: the sides of the backbox. The leftover piece should be about 22" x 48" (or less than 22" if you're building a widebody or wider-than-standard-body cabinet). Mark line "J" by measuring 28-1/2" from one edge. Cut along line "J".

12. On the 28-1/2" piece we just cut, mark the diagonal line "K" by measuring a point 10" from a 28-1/2"-long edge at one end, and 6-1/2" from the same edge at the other end. Cut along this line. This yields the first backbox side piece.

13. On the other piece, you can either repeat the measurement to mark the square line "L", or you can use the first backbox side piece as a template to mark the cut line, since the two sides are identical. Cut along the line.

We're finished with the first plywood piece! Time to move on to the second sheet.



Plywood sheet #2 - 4' x 8' x 3/4" nominal thickness.

14. These pieces will form the top and bottom of the backbox. Start by cutting line "A", at 28" from one edge.

You can cut along line "A" all the way across the board if you like, but as you can see, we only need the top corner for these pieces. If you want to keep more of the rest of the board intact as a large leftover piece, you can just cut about 18" in along line "A".

15. Mark line "B", at 10" from the outer edge. **Set your blade to a 7° tilt.**

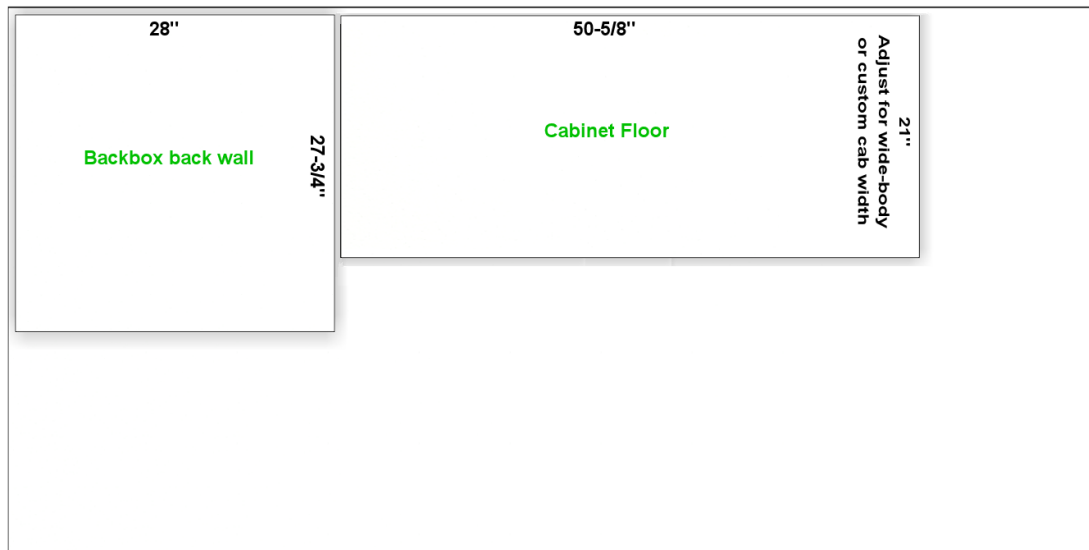
The blade should be tilted so that the face you're cutting into will be the wider side. If your saw only tilts the other way, so that the opposite face will be the wider one after the cut, flip everything around and measure line "B" at 6-1/2" in instead of 10" in.

Once you have it set up, cut at line "B" with the 7° tilted blade.

This yields the backbox top (or the backbox bottom, if you had to flip things around for the 6-1/2" cut).

Sanity check on the angled cut: The result should be 10" side on one face, and about 9-29/32" wide on the opposite face. If you did the 6-1/2" cut, the result should be 6-1/2" wide on one face. and slightly wider, about 6-19/32", on the opposite face.

16. **Set your saw back to 0° for square cuts.** Orient the remaining piece so that the angled cut with the narrow face is facing the saw blade. Measure line "C" at 6-1/2" from the angled edge (or 10" from the angled edge if you flipped things around in the previous step). Cut along this line. Make the same sanity check as in the previous step for the angled cut.



Plywood sheet #3 - 4' x 8' x 1/2" nominal thickness.

17. Now we're going to switch to 1/2" plywood (or particle board). The first cut is the backbox back wall. This is a simple rectangular piece, 28" x 27-3/4".

18. The second cut is the cabinet floor, another simple rectangular piece, 50-5/8" x 21".

If you're building a widebody or custom cabinet size, adjust the size to match. Here's how I calculate the floor size: start with the outside dimensions of your cabinet, then subtract 7/8" from each dimension. This accounts for the thickness of the walls left over outside of the dado grooves that the floor fits into (about 3/8" on each side, for 3/4" total), with another 1/8" of wiggle room, in case of any irregularities in the dado depth or floor edge. You can add back the 1/8" if you want a tighter fit; you can always do a test fit and sand it down a little if necessary.

19. There are a few more miscellaneous pieces that you'll need to cut out of the leftover plywood. These are covered in detail in Chapter 21, Cabinet Body, but here's a quick summary:

- Two corner supports for the cashbox fence, each 3" long, with a triangular cross section (exact sizing is unimportant); these can be made from a nominal 2x2 board cut in half diagonally (at 45°) lengthwise
- Two corner braces for the front leg brackets, each 6" to 8½" long (at your discretion), cut in a triangular cross-section (with two 1-1/16" sides and a 1-1/2" hypotenuse)
- Two corner braces for the back leg brackets, each 6" to 21½" long (at your

discretion), cut in a triangular cross-section (same as the front corner braces)

- Two 4-3/4" x 3/4" strips of 1/2" plywood, for DMD panel guides
- Two 15" x 3/4" strips of 1/2" plywood, for translite guides
- One 27-1/8" x 3/4" strips of 3/4" plywood, for a translite guide
- Two 12-3/8" x 1" strips of 3/4" plywood, for translite guides
- One 27-1/8" length of 3/4" reducer molding (or a similar shape fashioned from a 1x2), for backbox trim at the top of the translite

The woodworking on the plywood pieces isn't finished after you cut the last piece. Most of these parts require some additional work with a router, drill, and/or jigsaw. This is all covered in detail in Chapter 21, Cabinet Body.

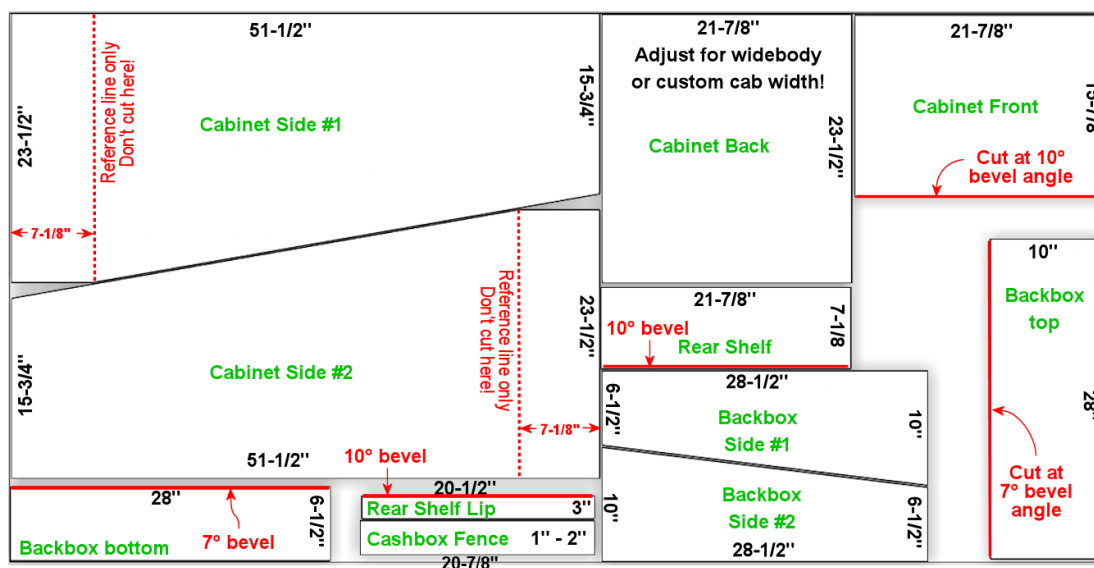
For help with the triangular wedge-shaped pieces, see Appendix 14, How to Make Corner Braces (and other wood prism shapes).

Layout 2: Single 3/4" sheet plan

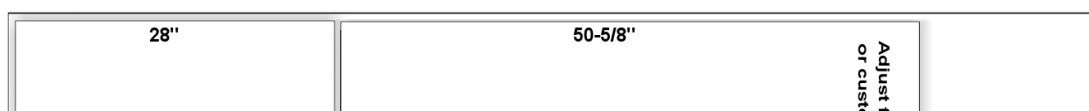
It's just barely possible to make all of the 3/4" pieces fit into a single 4x8 sheet. Plywood's not cheap, so this plan is easier on the budget. But this plan requires more measuring and cutting work than the group-by-size plan above, because it's not possible to group the pieces as nicely given the more limited space. Also, there's not enough spare room to allow for expanding any of the pieces to widebody widths, or to any custom width larger than the standard-body design. This plan will really only work for a standard-body build.

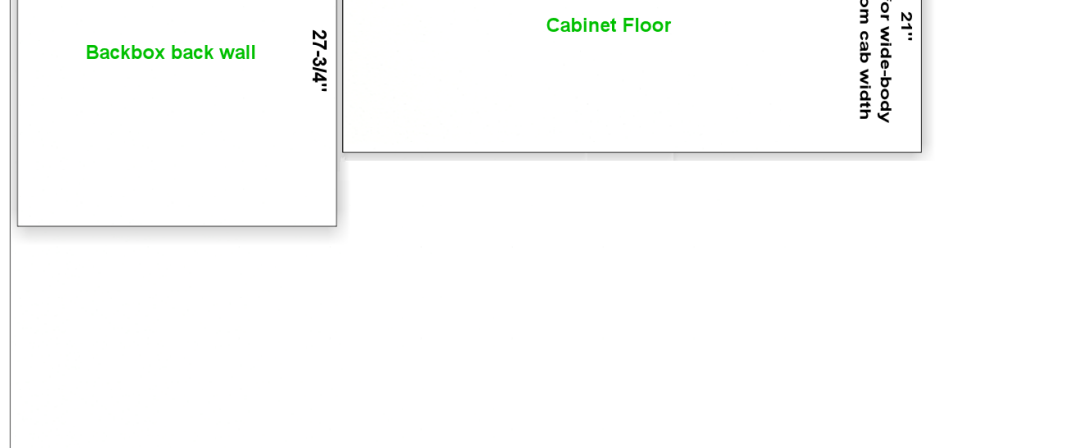
You'll have to be very careful with this plan to minimize wasted material between cuts, since everything is packed so tightly.

Note that "single" refers only to the 3/4" material. You do still need a second sheet, of 1/2" material, for the cabinet floor and backbox back wall.



*Plywood sheet #1 - 4' x 8' x 3/4" nominal thickness. Dimensions are all for the WPC standard-body design. For the beveled cuts, the dimensions are all given for the **longer** face on the resulting piece.*





Plywood sheet #2 - 4' x 8' x 1/2" nominal thickness.

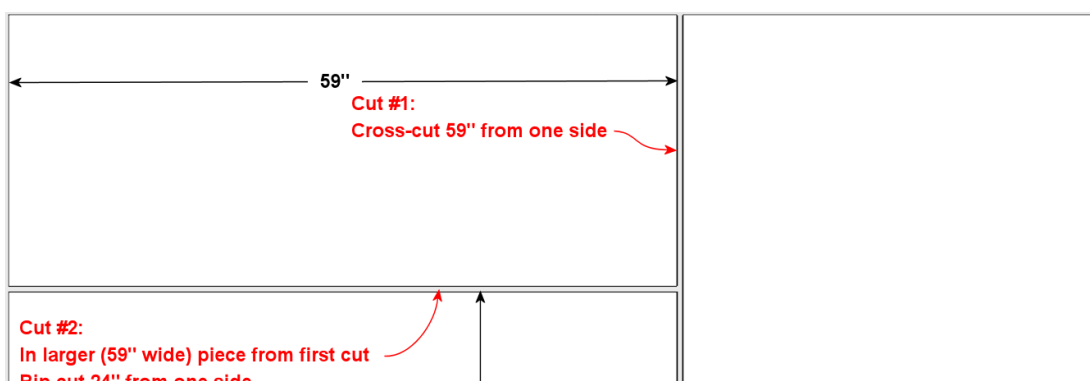
Layout 3: Car-friendly plan

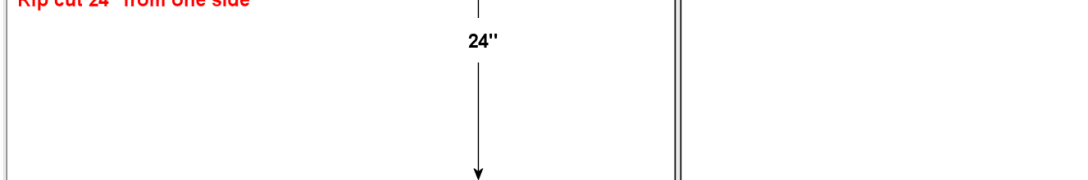
I don't own a pickup truck, so it's always a huge hassle for me to transport large sheets of plywood. One easy workaround is to ask the store to break down the sheets into smaller pieces that I can fit into my car. This is a free service at most Home Depot and Lowe's locations - they'll usually cut full plywood sheets into two or three pieces for you at no added charge. There are some caveats; they don't guarantee that the measurements will be exact, and they'll only do straight cuts parallel to the edges of the board. They also warn you that the big industrial panel saws they use might not leave a very clean edge. They don't consider this a "finish" carpentry service, just a convenience for easier loading.

As an example, I came up with the layout below to fit my car. You might be able to use this directly if it fits your car as well, but more likely you'll need to adapt it for your car's cargo area size. There are two rules you need to observe when creating your own layout. First, leave some dead space on each side of the store cuts - ideally about 1/2" on each side of each cut. That leaves margin for error in case the store is a little off with their measurements, and it lets you trim the edge more cleanly if their saw leaves a rough edge. Second, when you figure which pieces you can fit into a given area, leave a similar dead zone between pieces to account for your own saw blade's width - about 1/4" spacing between adjacent pieces should be sufficient.

This layout has an interesting bonus feature: if you happen to be building *two* pin cabs at once, there's enough leftover material that you can build the second cab with only one addition sheet of 3/4" plywood. (I don't think there's a way to avoid the need for a second sheet of 1/2" ply for the second cab, though, or at least another half sheet.)

My layout requires two pieces of 3/4" plywood plus one piece of 1/2" plywood. At the store, ask them to cut up each piece like this:





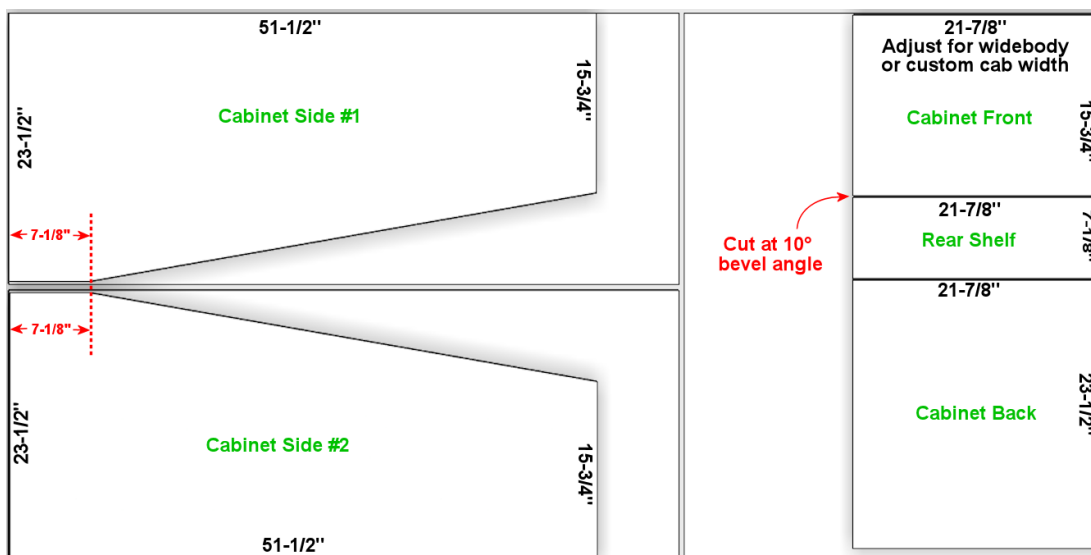
These are the cuts to make at the lumber store, to break down the sheets into smaller pieces for transport. The plan requires two sheets of 3/4" plywood plus one sheet of 1/2" material. You can have the store cut all three sheets the same way.

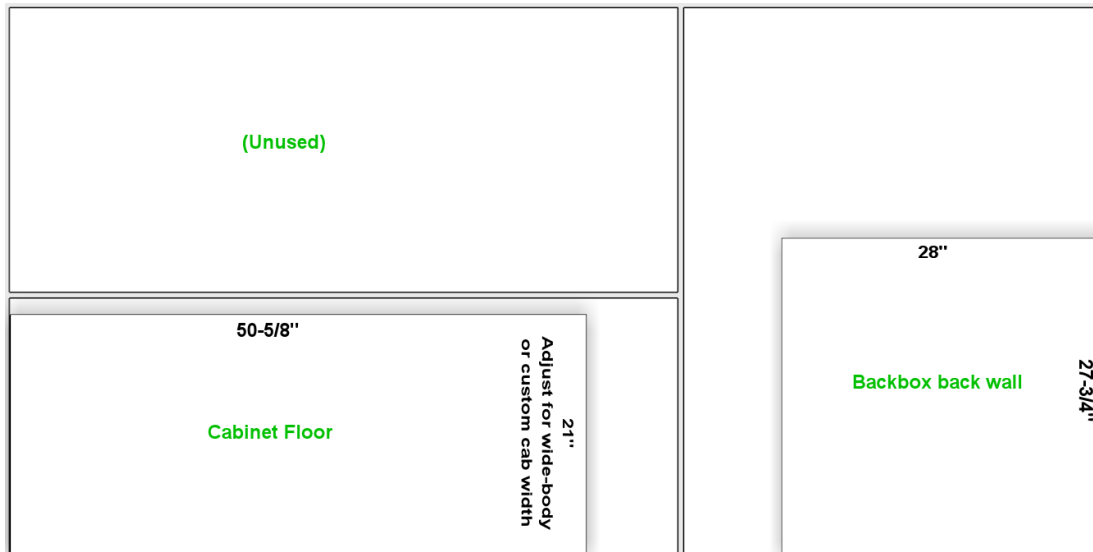
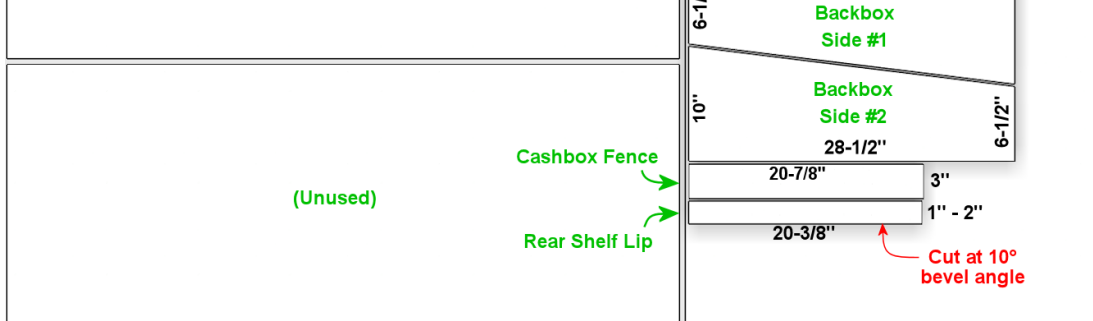
1. Grab two 4'x8'x3/4" plywood sheets and one 4'x8'x1/2" plywood sheet, and take them to the panel saw station
2. Ask the store associate to make a cross-cut in each sheet at 59" from one edge
3. Now ask the associate to make a rip cut in each of the **larger** (59" wide) pieces, at 24" - in other words, cut it in half length-wise

Note: The 59" measurement is based on making the smaller side of this cut as wide as I can fit in my car. The second piece ends up being about 37"x48" with this cut, and I can fit about 37" across my cargo area. If your car can accommodate a piece wider than 37", you can make more efficient use of the plywood by reducing the 59" measurement - but don't go below 52", since that part is for the cabinet side panels, which are 51.5" long. By the same token, if 37" is too wide for your car, you can increase the 59" measurement in order to make the other side narrower, assuming your car can fit a piece longer than 59".

Attention wide-body or custom width cabinet builders: You might have to change the final rip cut for the 1/2"-thickness sheet! Figure out how wide your cabinet is going to be on the outside, and make the final cut at that width or wider, **instead of** the 24" measurement above.

Here's how I mapped the cabinet panels onto the broken-down plywood sheets:





Note that you don't need a full second 4'x8' sheet of the 3/4" plywood - a half-sheet would be perfectly adequate, if your lumber store offers it. A half-sheet isn't workable for the 1/2" section, unfortunately, since the cabinet floor piece is longer than 48".

Appendix 10. Cabinet Construction Quick Reference

This section is a quick reference to the woodworking procedure to construct the main cabinet shell, based on the Williams/Bally standard-width cabinets of the 1990s (which is also used more or less unchanged by most newer commercial machines). The plan here includes plywood cutting dimensions and locations for drills, cutouts, and routed grooves, and it's organized into a step-by-step procedure that you can follow.

This is the same plan that's laid out in Chapter 21, Cabinet Body, but it's distilled here to just the essential checklist steps, arranged into an efficient order of execution. The order of steps makes sure that dependencies are done by the time you need them, and groups operations that use similar tool setups, as much as possible, to minimize the time you have to spend changing drill bits and so forth. The goal is that you can just go down this list while doing the woodworking and carry out the steps in order. To keep it concise, there's not much explanation here, so I've provided links at each step to the more detailed corresponding material in the main Cabinet Body chapter.

This plan is primarily for virtual cab builders, but it can also be used to build replacement cabs for WPC-era machines and later System 11 machines, with some small variations. I've tried to point out in the text where changes are needed for replica cabs.

Corner joinery

The plans below don't specify the type of corner joins to use for the main cabinet, so they don't include any routing for those joins. You can do the joinery a number of different ways, each with its own advantages, so that's left to your discretion. However, the measurements assume that all of the pieces extend all the way to the corners of the finished cabinet, which is only true when you're using one of the mitered joins (such as a simple 45° miter, a locking miter, or a mitered rabbet). If you're using non-mitered joins, such as butt or rabbet joins, you'll have to adjust the measurements for the pieces that don't extend all the way to the finished corners. References:

- Appendix 12, Lock Miter I: The Plywood-Friendly Way - this is the join used in the original WPC cabinets
- Appendix 13, Lock Miter II: The Special Router Bit Way
- Chapter 21, Cab Body - Joinery

The backbox plans do include specific joinery routing, which is based on locking rabbet joins.

Width adjustments for different cabinet types

These plans are based on the WPC "standard-body" size, which has an inside width of 20.5" in the main cabinet. Williams also made "widebody" cabinets with an inside width of 23.25". You can also choose a custom width.

To adapt the plans to a different finished cabinet width, you need to adjust the widths of the following pieces:

- Front wall
- Back wall

- Rear shelf
- Cab floor

Reference: Chapter 21, Cab Body - Standard and Widebody cabinets

The backbox width is typically kept at the same standard size regardless of the cabinet width.

Mark the panels as you go

As you cut the panels, I think it's a good idea to mark each piece with the following information, to avoid confusion as you work on them later:

- Function of the piece ("cab left wall", "backbox top", etc.)
- **Inside** and **outside** face
- **Top** and **bottom** directions

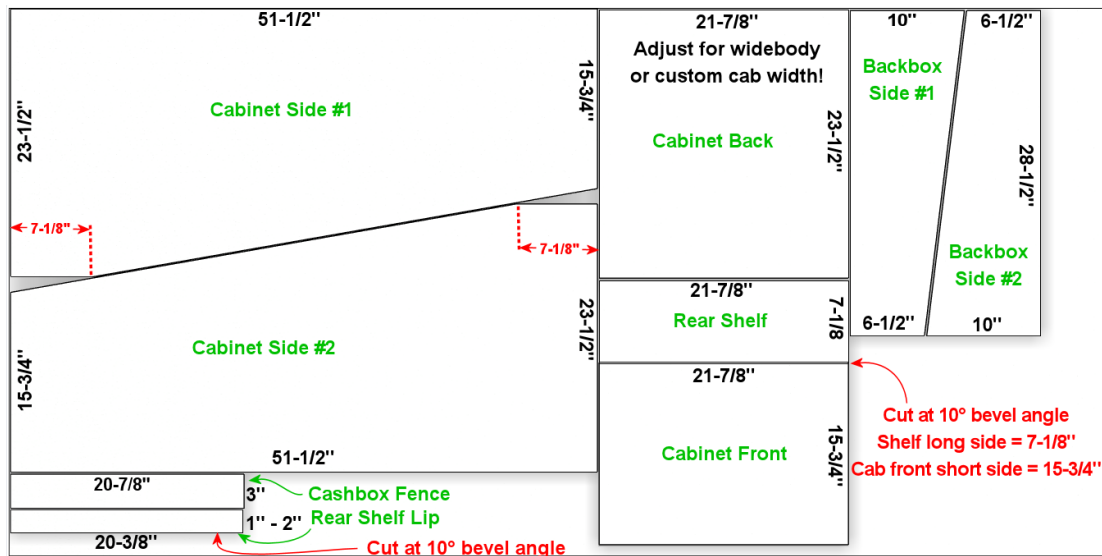
I just write all of this directly on each piece with a pencil.

Cut the panels

Reference: Appendix 9, Plywood Cutting Plans for Cabinet Construction

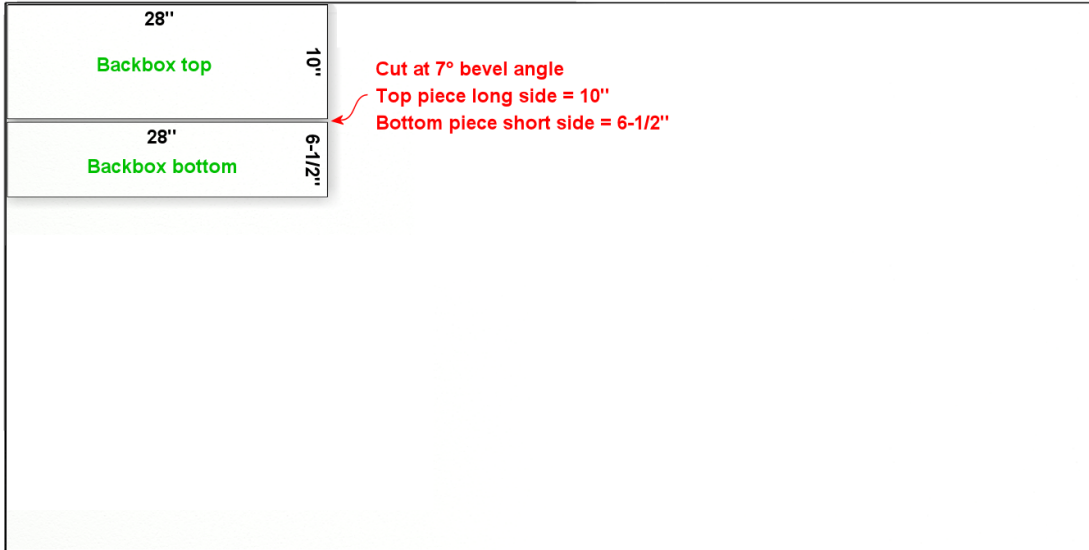
Note: There are other ways to divvy up the plywood, but for the sake of brevity, we're assuming the "grouped by size" layout here. See the chapter reference above for other layouts.

Plywood sheet #1 (4' x 8' x 3/4"):



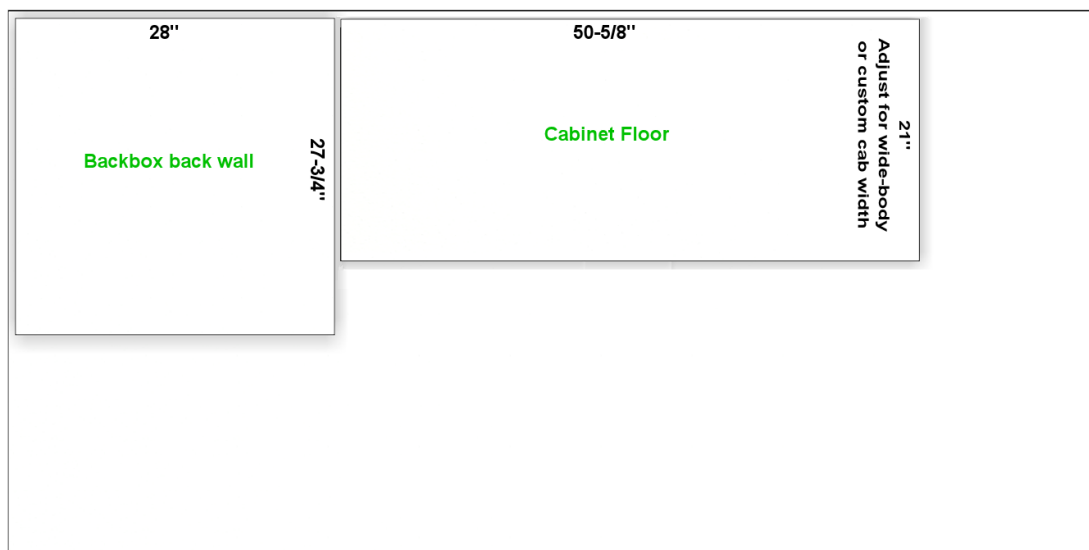
1. Cut the side pieces.
2. Cut the cab, front, and shelf pieces. **Note the 10° bevel angle** on the cut between the cabinet front and rear shelf - see Chapter 21, Cab Body - Front Wall. Adjust the width for widebody or custom widths.
3. Cut the backbox side panels.
4. Cut the cashbox fence and shelf lip.

Plywood sheet #2 (4' x 8' x 3/4"):



5. Cut the backbox top and bottom pieces. **Note the 7° bevel angle** on the cut between the top and bottom - see Chapter 21, Cab Body - Backbox top and Chapter 21, Cab Body - Backbox floor.

Plywood sheet #3 (4' x 8' x 1/2")



6. Cut the backbox back wall, 28" wide by 27-3/4" tall.

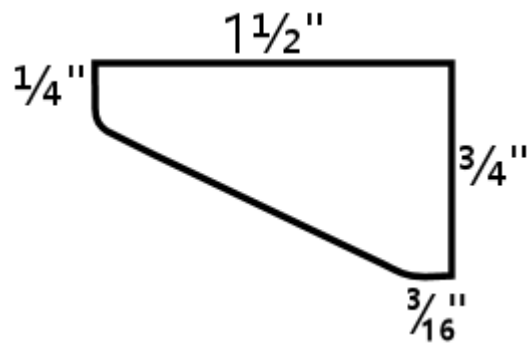
7. Cut the cabinet floor - adjust if you're building a widebody or custom-size cabinet.

Cut the backbox trim pieces

Quantity	Material	Dimensions
2	1/2" plywood	4 3/4" x 3/4"
2	1/2" plywood	15" x 3/4"
1	3/4" plywood	27 1/8" x 3/4"
2	3/4" plywood	12 3/8" x 1"

1	1x2 board cut to $\frac{3}{4}$ " reducer molding shape (see diagram below)	27 $\frac{1}{8}$ " length
---	--	---------------------------

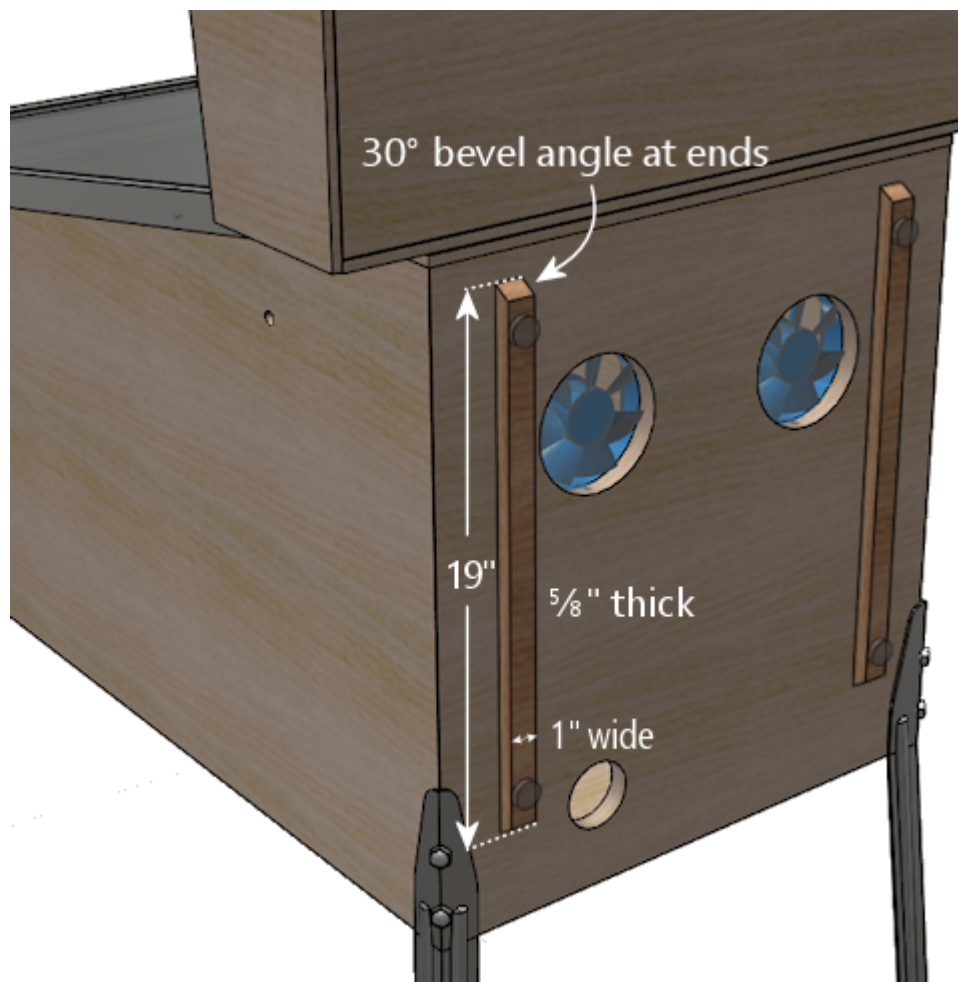
Reducer molding cross section:



Reference: Chapter 21, Cab Body - Translite and DMD guides

Cut the main cab back rails

Two pieces of $\frac{5}{8}$ " thick plywood, 19" long by 1" wide. Cut ends at a 30° bevel angle. (Substitute $\frac{3}{4}$ " plywood if desired.)



Reference: Chapter 21, Cab Body - Back rails

Cut the corner braces

Cut a 2x2 board diagonally down its length to form a triangular prism shape, for corner bracing under the leg brackets and to attach the cashbox fence. Optionally, you can use similar pieces to reinforce the joints between the floor and side walls. (Don't use them with the floor joints with the front and back walls, though, since those might get in the way of other hardware.)

For the front and back corners, the following profile is required to fit the space under the leg brackets:



Lengths:

- Front corners (leg bracket profile): 6" to 8-1/2" long, quantity 2
- Back corners (leg bracket profile): 6" to 21-1/2" long, quantity 2
- Cashbox fence: 3" long, quantity 2
- Floor braces along side walls (optional): up to 30" long, quantity 2

References:

- Chapter 21, Cab Body - Cashbox fence
- Chapter 21, Cab Bod - Leg bolt braces
- Appendix 14, How to Make Corner Braces (and other wood prism shapes)

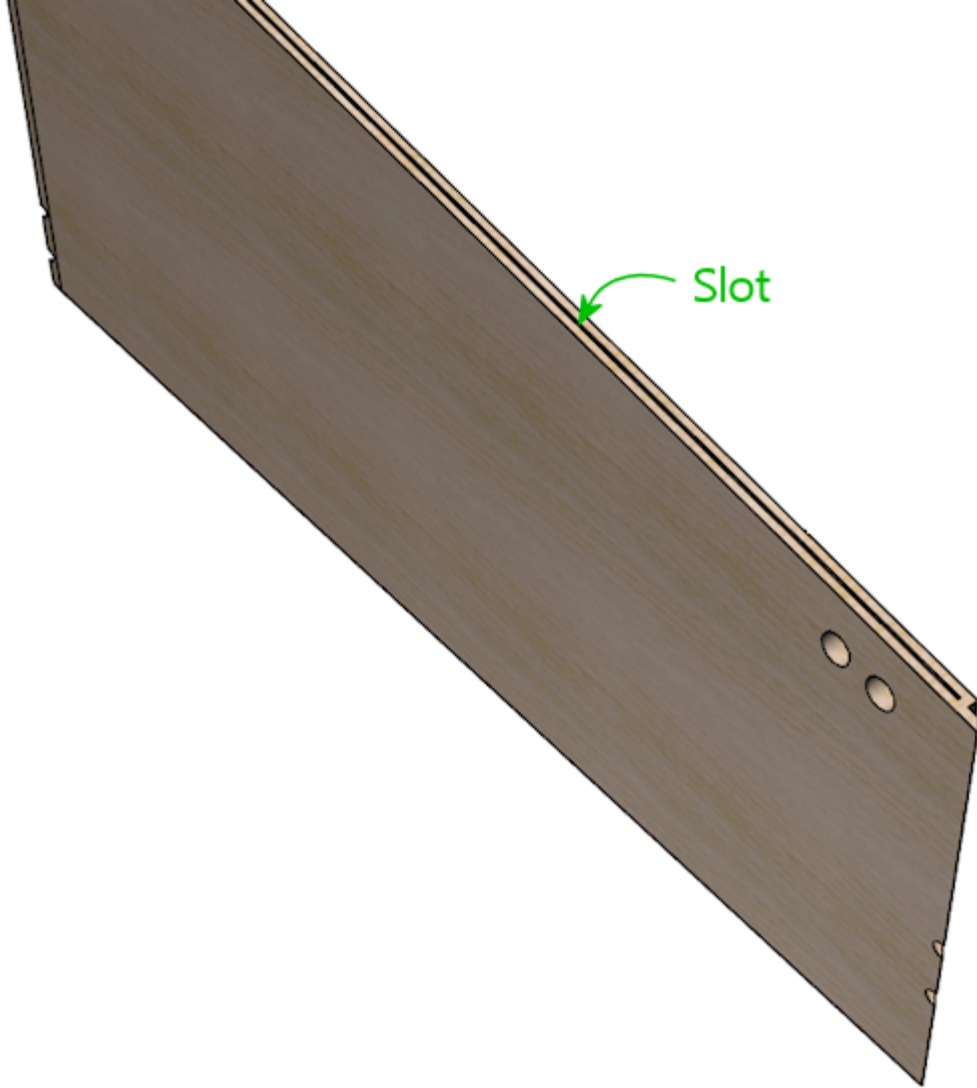
Route the top glass guide slots

Use a slot cutter router bit, 3/32" slot width, 3/8" slot depth (Freude part #63-106 or equivalent).

Route the top edge of each side panel, along the slanted section, centered in the edge. Route from about 1" from the front to the top of the slanted section.

Reference: Chapter 21, Cab Body - Glass channel slots





Mark all routing & drilling locations

At this point, I like to mark all of the panels with the locations of the routed grooves, cutouts, and drills - essentially everything below. It's easier to make many of the measurements now, while the pieces are still whole, and having everything pre-marked makes the execution much easier.

Cut the corner joins for the main cabinet

Route or cut the grooves and miters for your selected style of corner joins for the main cabinet. This applies to the main cabinet side, front, and back walls, at all four corners.

(I prefer to cut the corner joins first, before the floor dados, so that the dados don't get in the way when you're aligning the cuts for the corner joins.)

References:

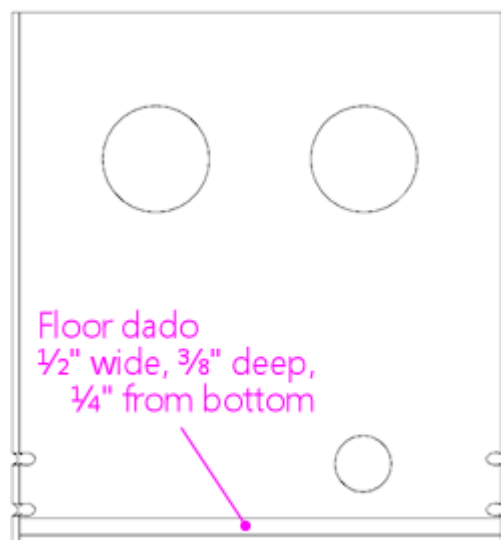
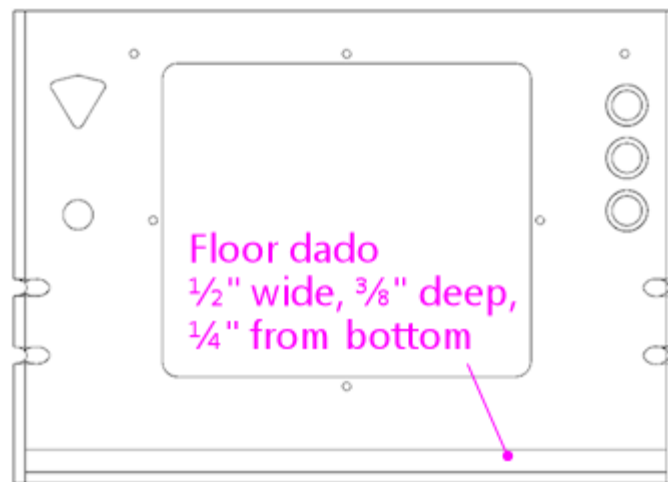
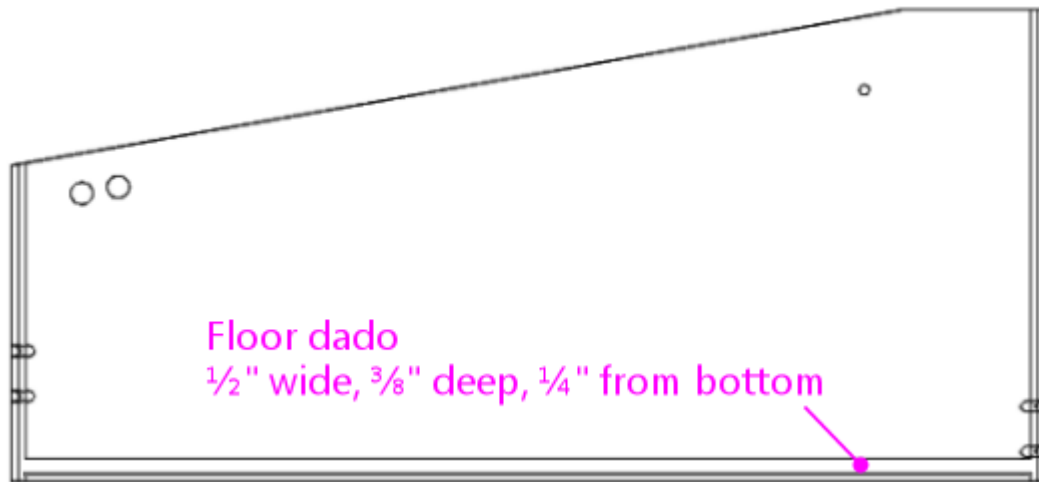
- Chapter 21, Cabinet Body - Joinery
- Appendix 12, Lock Miter I: The Plywood-Friendly Way
- Appendix 13, Lock Miter II: The Special Router Bit Way

Route floor dados

Cabinet sides, front, and back - inside faces.

Route 1/2" wide by 3/8" deep, offset 1/4" from bottom edge.

Note: Some of the 1990s machines I've checked have bottom offsets closer to 3/8".

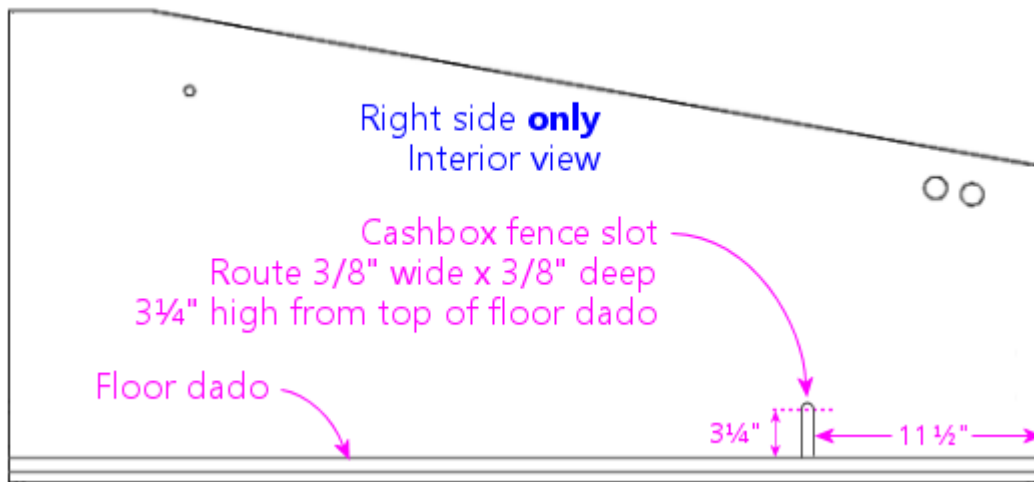


References:

- Chapter 21, Cab Body - Side walls
- Chapter 21, Cab Body - Front wall
- Chapter 21, Cab Body - Back wall

Route the cashbox fence slot

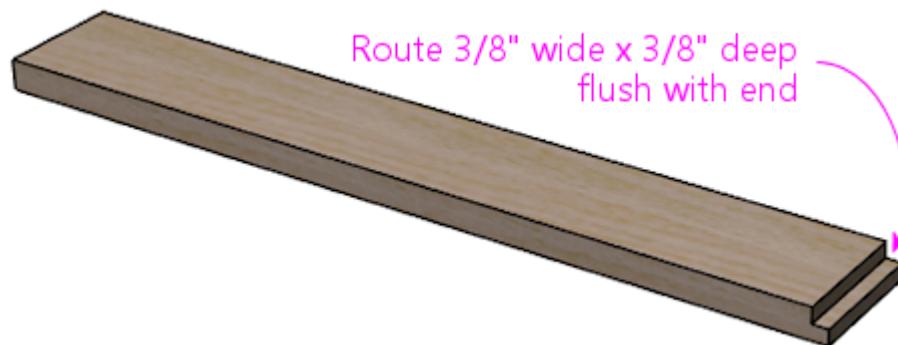
In the **right wall only**, on the **interior face**, route a slot $\frac{3}{8}$ " wide by $\frac{3}{8}$ " deep by $3\frac{1}{4}$ " high, with the bottom at the floor dado, and the front edge $11\frac{1}{2}$ " from the front (the outside corner).



Reference: Chapter 21, Cab Body - Side walls

Route the cashbox fence locking tab

Route a groove in the cashbox fence, $\frac{3}{8}$ " wide by $\frac{3}{8}$ " deep, flush with one end.

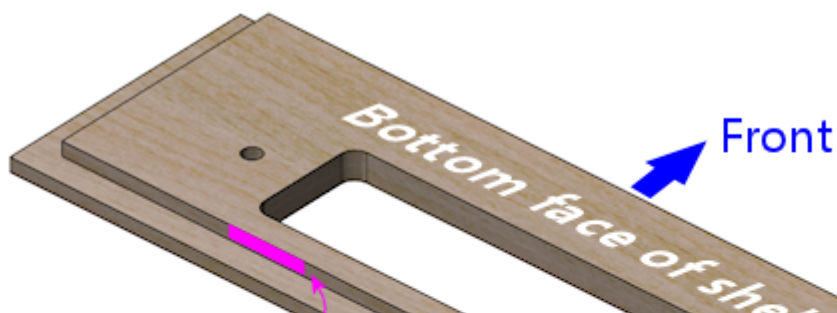


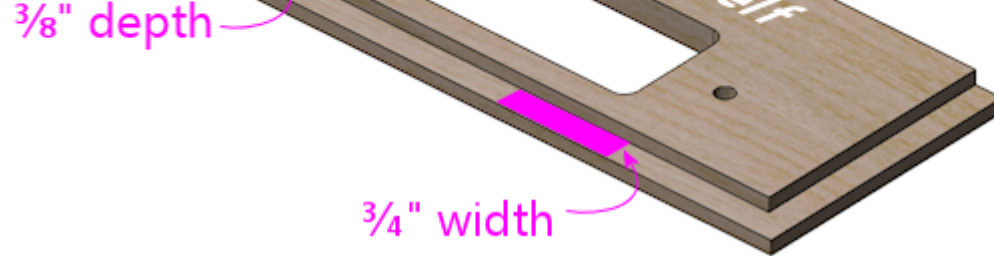
Reference: Chapter 21, Cab Body - Cashbox fence

Route shelf bottom dados

Shelf, bottom side

Route $\frac{3}{8}$ " wide by $\frac{3}{8}$ " deep grooves at the back, left, and right edges, flush with the edges

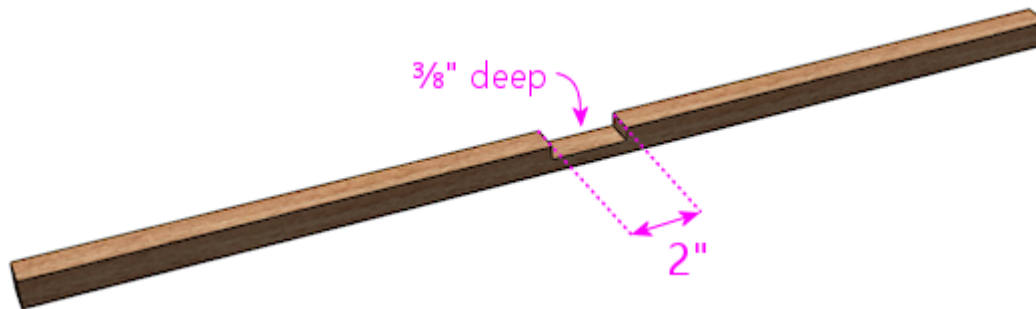




Reference: Chapter 21, Cab Body - Rear shelf

Route the translite lock cutout in the backbox top trim

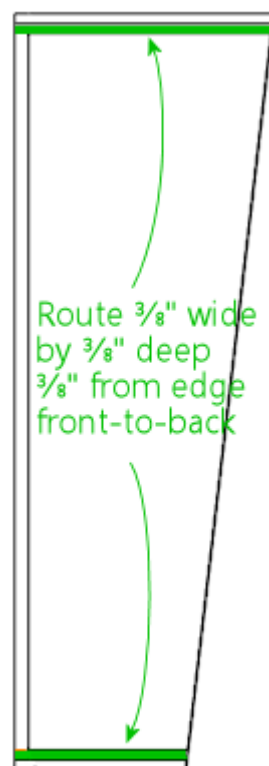
In the 27-1/8" x 3/4" x 3/4" backbox top trim piece: route a 2" wide inset, 3/8" deep, centered side-to-side. (Ignore this step if you're not planning to install a translite lock.)



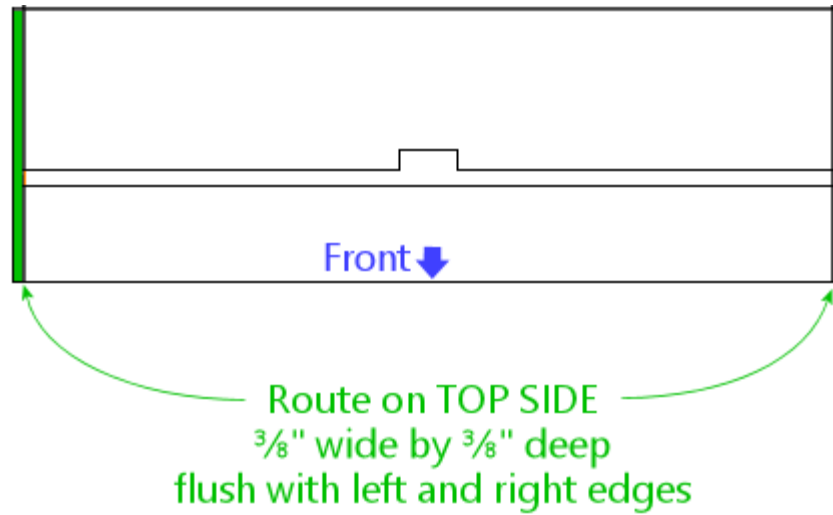
Reference: Chapter 21, Cab Body - Extra routing for translite lock

Route the backbox corner joins

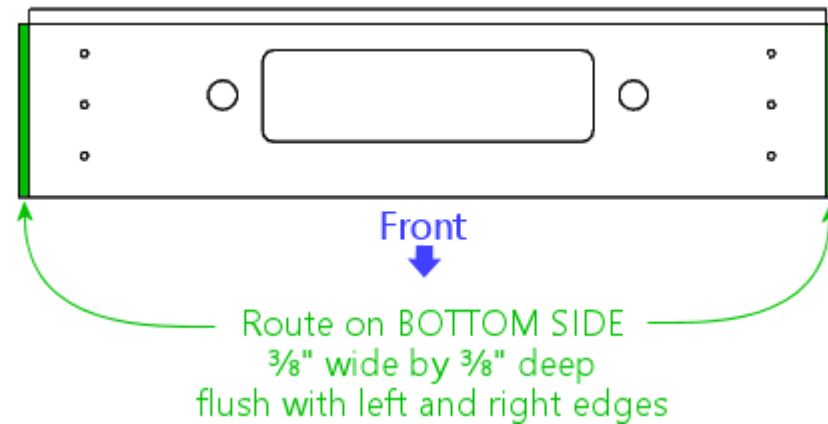
Left and right backbox side walls: Route grooves on **inside** faces, 3/8" wide by 3/8" deep, offset 3/8" from the top and bottom edges.



Backbox top: Route on **outside** (top) face, 3/8" wide by 3/8" deep, flush with left and right edges.



Backbox bottom: Route on **outside** (bottom) face, 3/8" wide by 3/8" deep, flush with left and right edges.



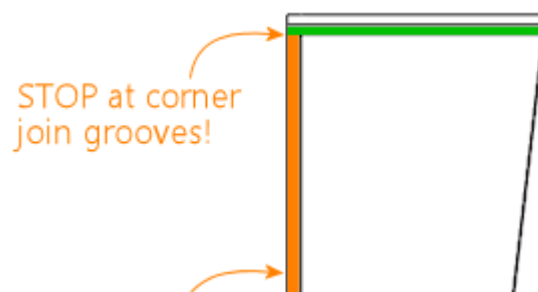
References:

- Chapter 21, Cab Body - Backbox sides
- Chapter 21, Cab Body - Backbox top
- Chapter 21, Cab Body - Backbox bottom

Route the backbox back wall grooves

Backbox sides, **inside** faces: Route 1/2" wide by 3/8" deep, flush with the back.

Important: Only route **between** the top and bottom corner join grooves. Only route the orange section as shown below.

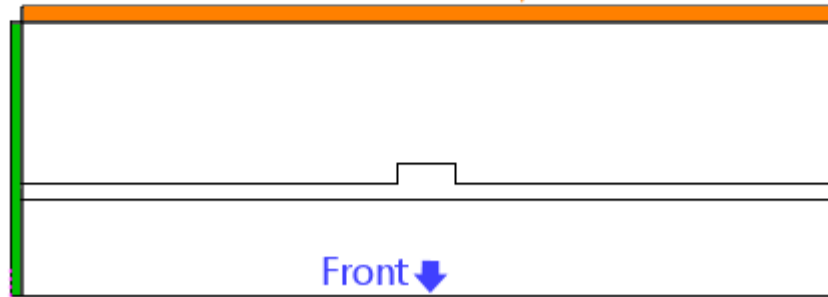


Route 1/2" wide
by 3/8" deep
to back edge

STOP at corner
join grooves!

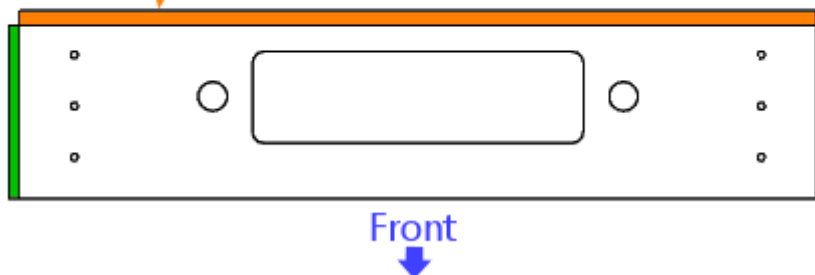
Inside Face

Route on BOTTOM SIDE
1/2" wide by 3/8" deep
flush with back edge



Backbox bottom, **inside** (top) face: Route 1/2" wide by 3/8" deep, flush with the back.

Route on TOP SIDE
1/2" wide by 3/8" deep
flush with back edge



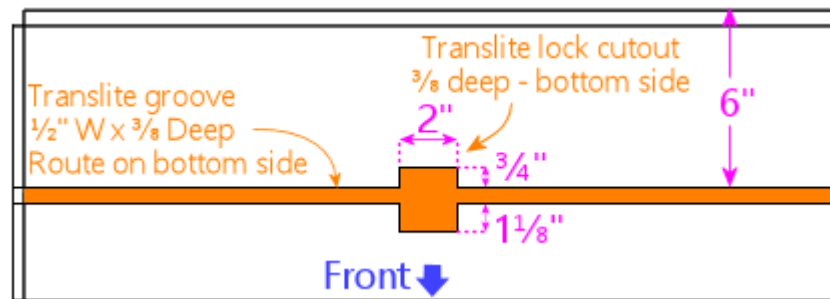
References:

- Chapter 21, Cab Body - Backbox sides
- Chapter 21, Cab Body - Backbox top
- Chapter 21, Cab Body - Backbox bottom

Route the translite grooves

Backbox top, **inside** (bottom) face:

- Route a groove, 1/2" wide by 3/8" deep, 6" from the back edge, across the whole width of the piece. (The translite fits into this recess.)
- Route a rectangular inset, 2" wide by 2-3/8" tall by 3/8" deep, as shown in the diagram below. (This is for the translite lock.)

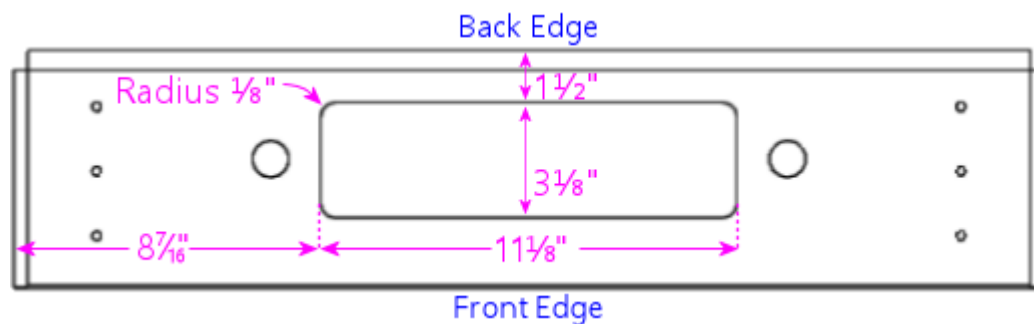


Reference: Chapter 21, Cab Body - Backbox bottom

Cut the backbox floor cable opening

Backbox bottom: Cut a rectangular opening, 11-1/8" by 3-1/8", as shown in the diagram.

Note: this lines up with the cable opening in the shelf when the back edges are aligned.

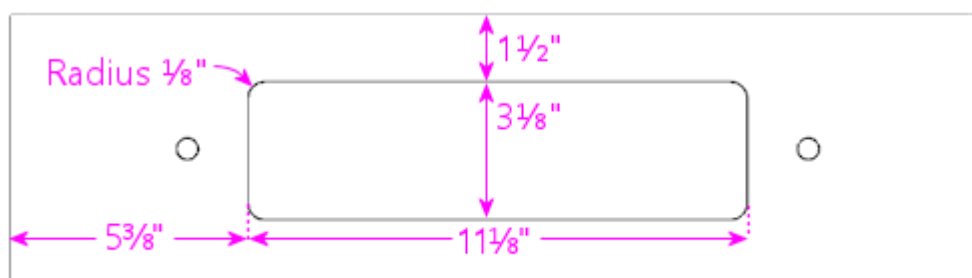


Reference: Chapter 21, Cab Body - Backbox bottom

Cut the shelf cable opening

Shelf: Cut a rectangular opening, 11-1/8" by 3-1/8", as shown in the diagram.

Note: this lines up with the cable opening in the bottom of the abckbox when the back edges are aligned.

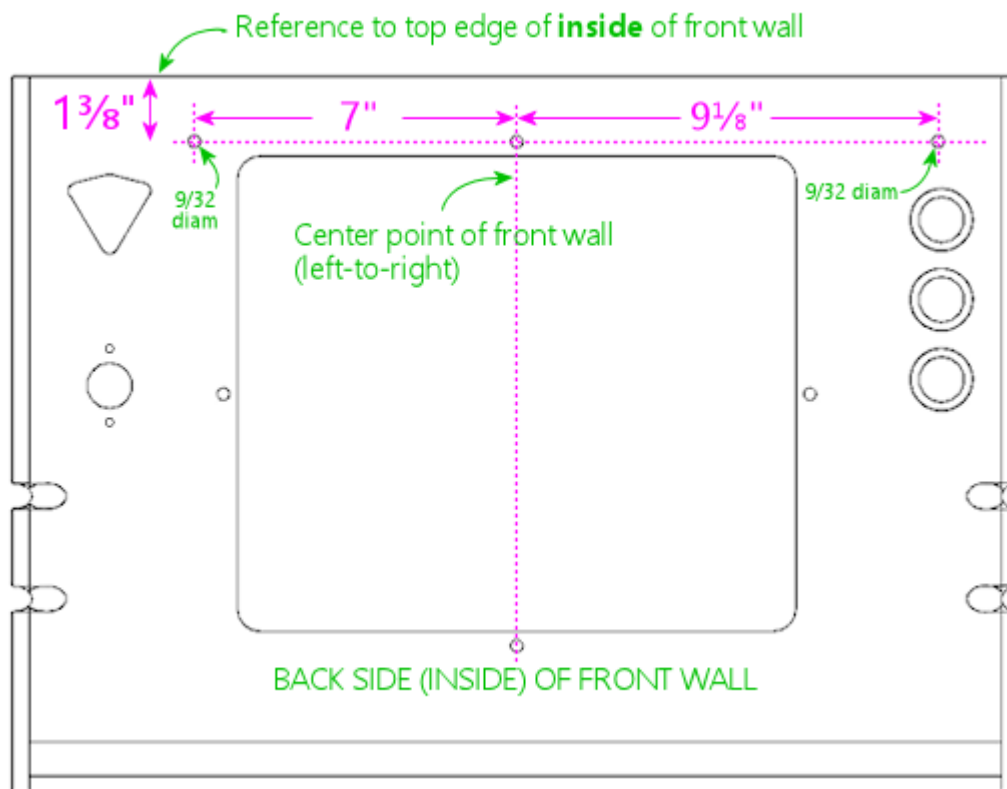


Reference: Chapter 21, Cab Body - Shelf

Test the lockbar fit

The measurements shown below for the coin door cutout and lockbar bolts are based on the standard equipment, but I've run into some slight manufacturing variations in the lockbar receivers, so I like to use the actual receiver I'm going to install as a template for drilling. That helps ensure that the final fit is closer to perfect.

Here's the procedure. Place your lockbar receiver against the **inside** face of the front panel, and align the two little tabs on the front so that they're exactly flush with the top edge of the panel. Mark the locations of the three bolt holes. Remove the receiver and measure the distance from the top edge of the panel to the center of the marked bolt hole locations. Compare to the diagram below:



If the position you mark by testing with the receiver differs by more than $\frac{1}{32}"$ vertically from the diagram above, I'd use the "test" positions instead of the diagram locations. You should also adjust all of the following up or down by the same amount to match, since all of these pieces fit together when installed:

- Coin door cutout
- Lockbar bolt holes
- Coin door bolt holes

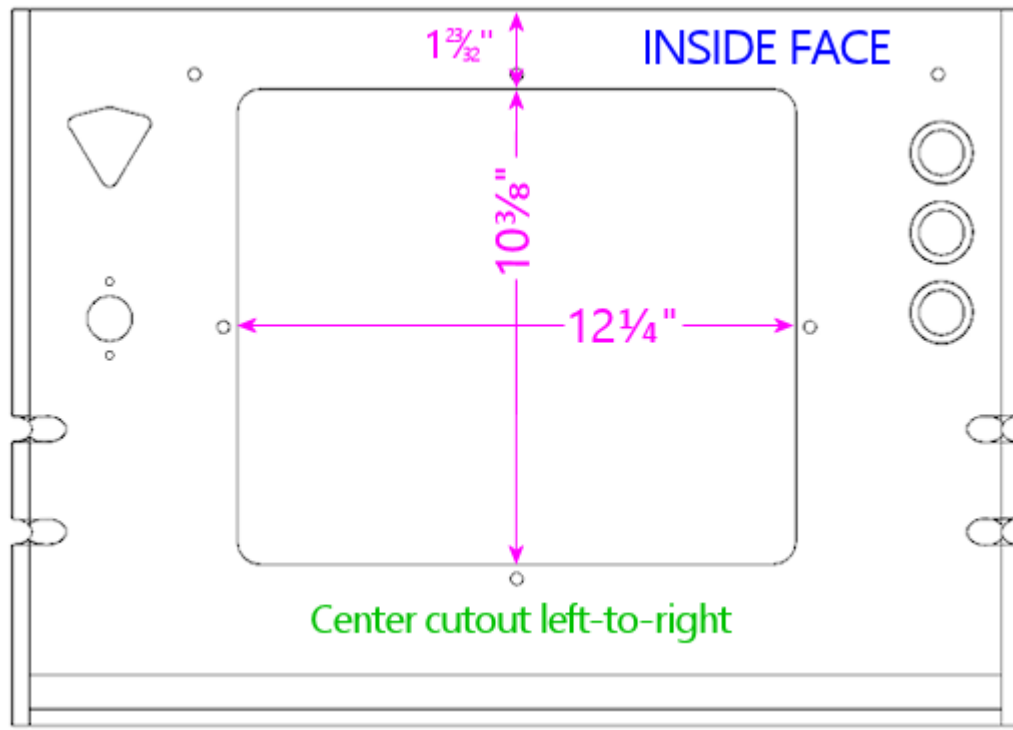
Note that the lockbar receiver is drilled to give you about $\frac{1}{4}"$ of play side-to-side, so the horizontal locations don't have to be as exact.

Cut the coin door opening

Note: see Test the lockbar fit above before proceeding.

Cut a rectangular opening, 12-1/4" wide by 10-3/8" high, centered left to right. The top of the cutout is 1-23/32" from the top of the **inside** face.

(The distance is specified from the top, because the coin door has to align with the lockbar receiver, which has to be a certain distance from the top for the lockbar to fit.)

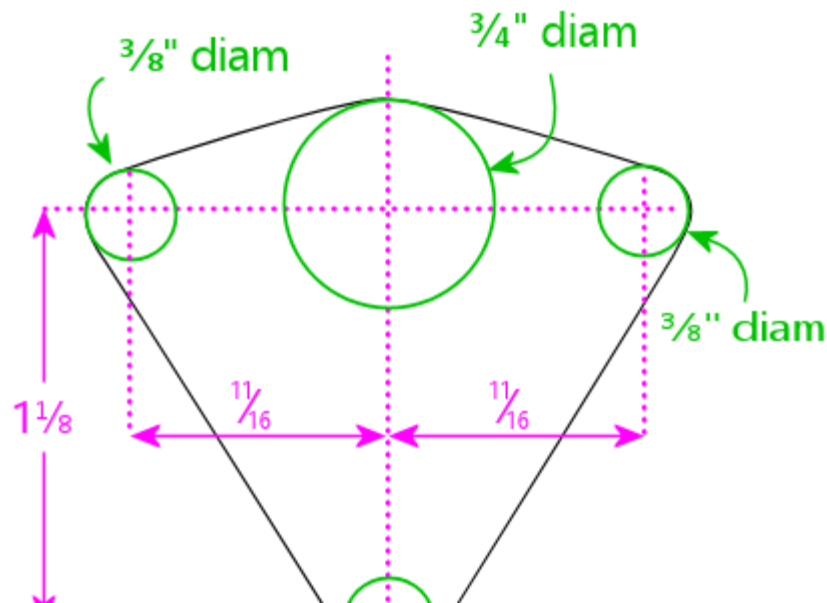


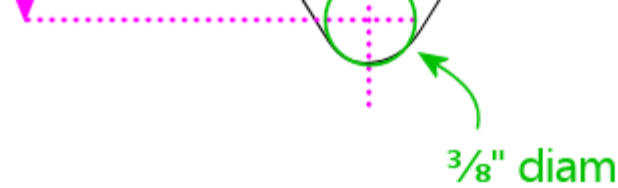
Reference: Chapter 21, Cab Body - Coin door cutout

Cut the plunger opening

Do this only if you're installing a plunger (ball shooter). This cutout shape only applies to the modern style, 1980s and later.

Cutout shape: Drill the four holes on the centers shown, then route or jigsaw along the perimeter they define (black outline in the diagram below).





Cutout location: Varies. For virtual cabs, a location similar to that used in real machines is preferable for aesthetics, but you can move it if needed to avoid space conflicts with the TV. For replacement pinball cabs, the location is strictly dictated by the playfield geometry, because the plunger has to line up with the playfield shooter lane. The table below lists typical locations for Williams games of the 1980s and 1990s, but individual titles may vary, so check against a factory original if possible.

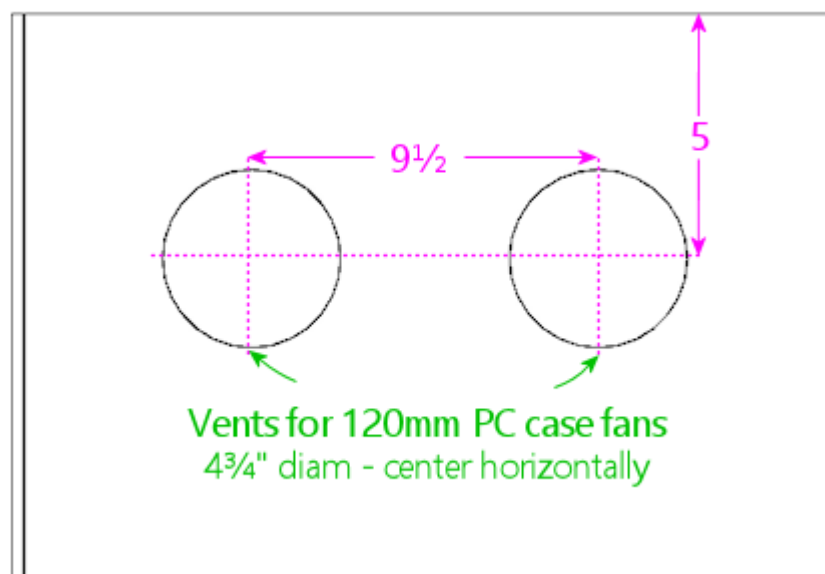
The reference point for all table entries below is the center of the top 3/4" diameter drill. The location is measured from the top and right edges of the **outside** face of the front panel.

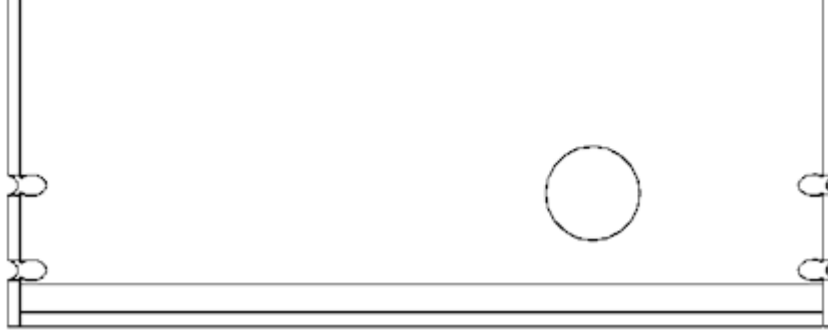
Cabinet Usage	Distance from top	Distance from right
Virtual pin cab - plunger only, or plunger above Launch Ball button	2-1/2"	2-1/8"
Virtual pin cab - plunger below Launch Ball button	5"	2-1/8"
Replacement cab for Williams System 11 and early WPC titles, through 1993	1-5/8"	2-1/8"
Replacement cab for later WPC titles, 1994 and later	2-1/2"	2-1/8"

Reference: Chapter 21, Cab Body - Plunger and Launch button

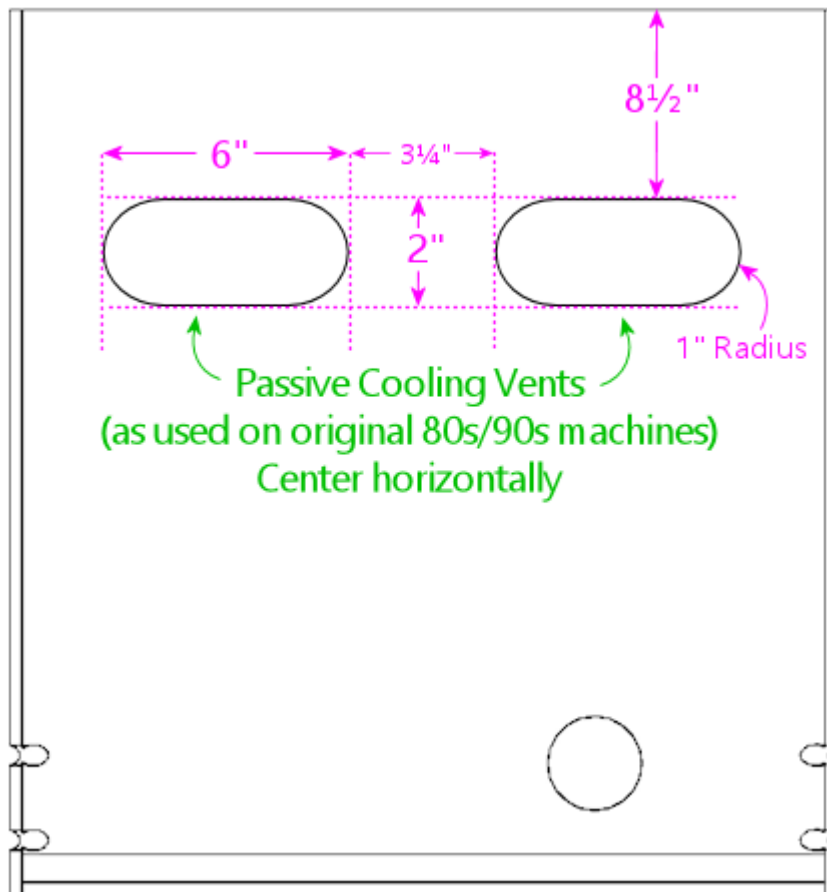
Cut the back wall vents

Virtual cabs: For 120mm PC case fans, cut 4-3/4" diameter circular openings. The exact location isn't critical, so adjust as desired.





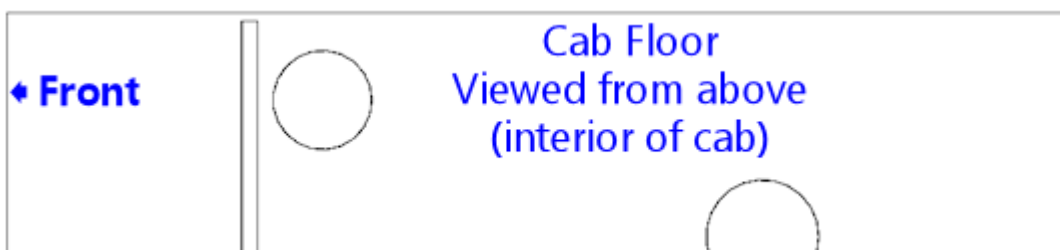
Replacement pinball cabs: Cut two passive vents, 6" wide by 2" high, with 1" radius rounded ends, as shown in the diagram below. (Use a 2"-diameter hole saw to drill the rounded ends, then cut the straight edges between the holes with a router or jigsaw.)

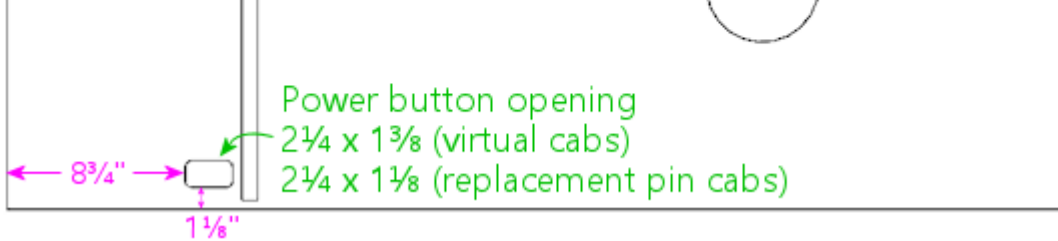


Reference: Chapter 21, Cab Body - Rear wall

Cut the power button opening

Cabinet floor: a rectangular opening, 2-1/4" long (in the long direction of the floor) by 1-3/8" wide (virtual cabs) or 1-1/8" wide (replacement WPC pinball cabs), at the location shown in the diagram.





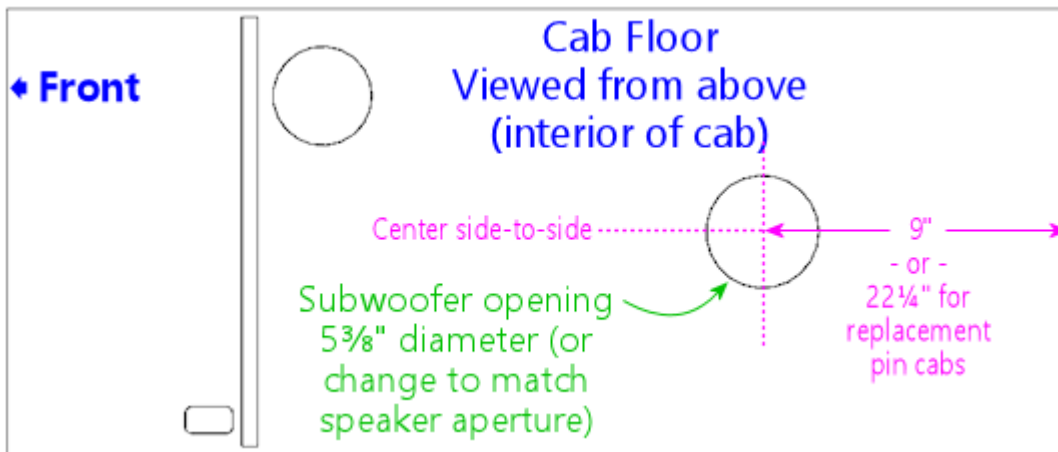
Reference: Chapter 21, Cab Body - Floor

Cut the subwoofer opening

Cabinet floor: circular opening, $5\text{-}3/8"$ diameter, centered side-to-side. Increase the diameter, if desired, to match your subwoofer's aperture.

For virtual cabs, the exact placement is up to you, but it's typically fairly close to the back, to leave a large block of space for the PC equipment. The diagram says 9" from the back, but this is just a suggestion.

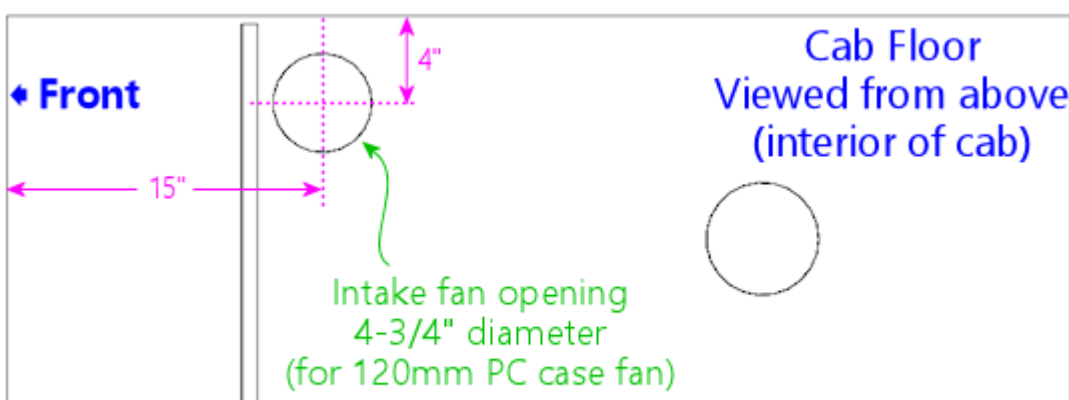
For replacement WPC cabs, the opening is typically at $22\text{-}1/4"$ from the back.



Reference: Chapter 21, Cab Body - Floor

Cut the floor intake fan opening

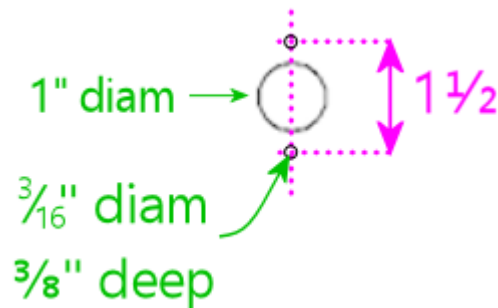
Virtual cabs only (omit for pinball replacement cabs): Cabinet floor, circular opening, sized to the intake fan (for a standard 120mm PC case fan, make it $4\text{-}3/4"$ diameter). There's no standard location. The diagram shows a possible location that should leave enough space for the PC motherboard. Some people also add a second intake fan for more air flow, mirrored on the opposite side.)



Drill the Launch Ball button

If you're using a Launch Ball button in addition to or instead of a plunger: In the front wall:

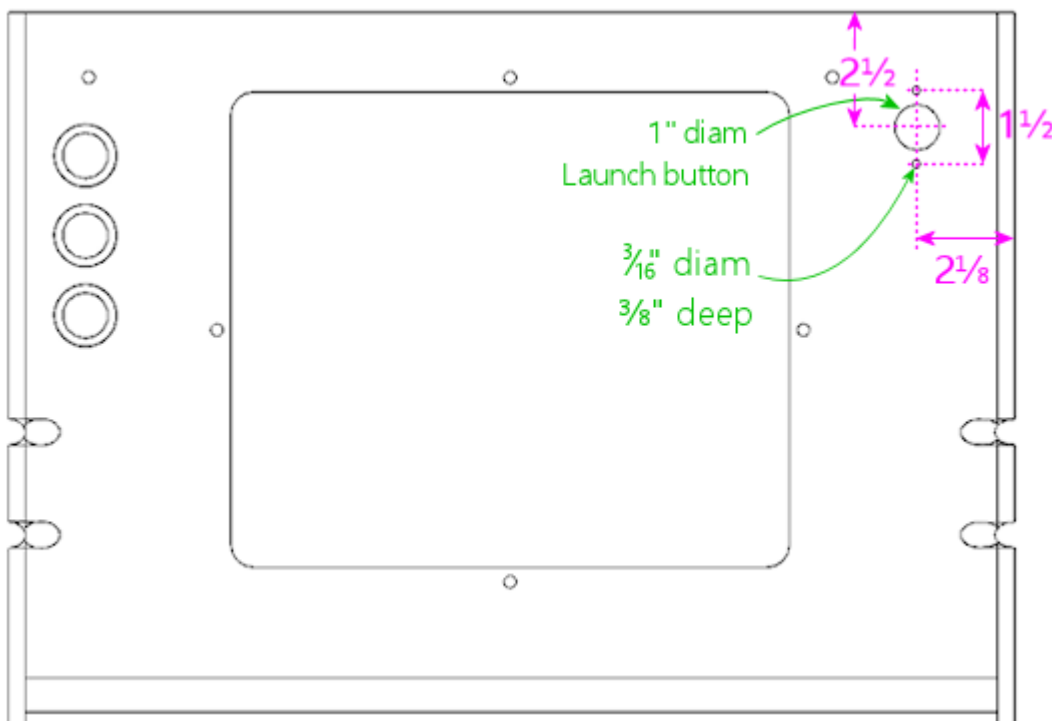
- Drill a 1" diameter hole, for the main shaft of the button
- Drill with two $\frac{3}{16}$ " diameter holes about $\frac{3}{8}$ " deep, spaced $1\frac{1}{2}$ " apart, one above and one below; these are for little nubs on the button housing that prevent it from rotating freely



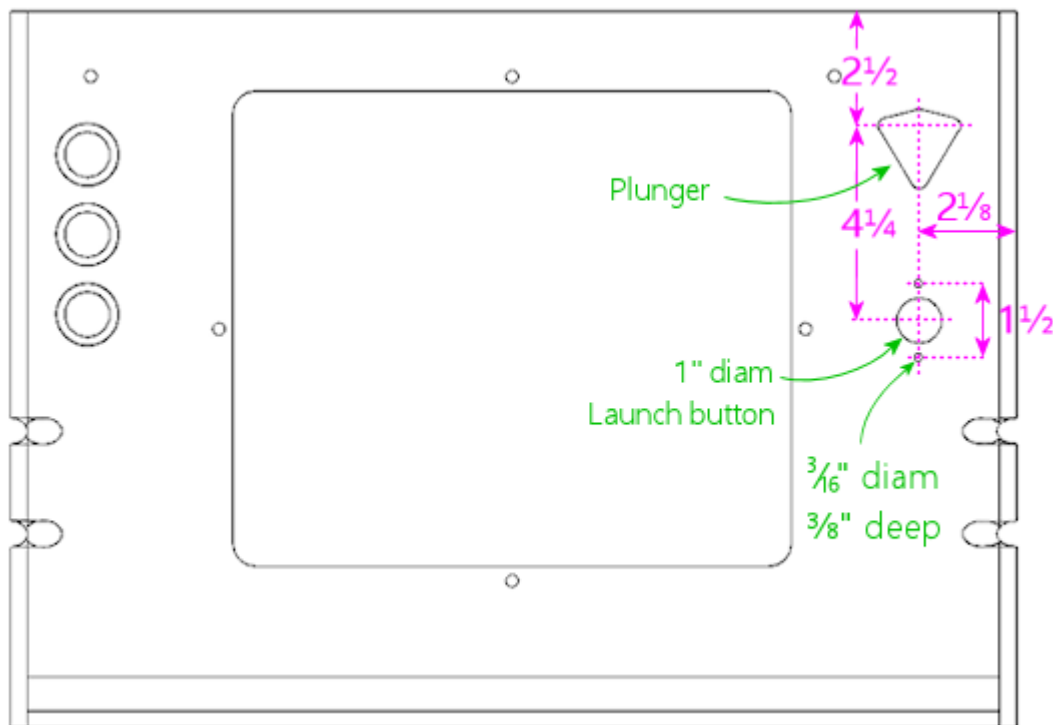
The location isn't critical, other than avoiding space conflicts with the lockbar receiver, leg bracket, and coin door. Typical placements:

- **Launch Ball button only (no plunger):** Place the button where the plunger would normally go, with the drill center $2\frac{1}{2}$ " from the top edge of the front wall, and $2\frac{1}{8}$ " from the right edge.

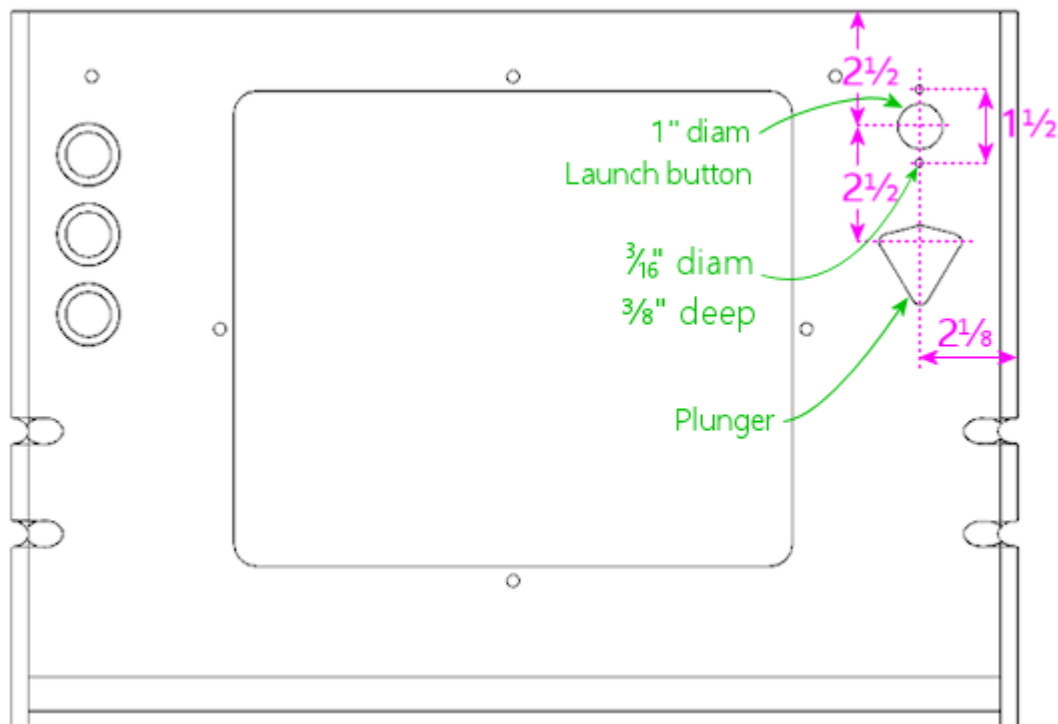
(This is suitable for replacement cabs for most WPC titles that used Launch Ball buttons instead of plungers.)



- **Plunger + Launch button, plunger on top:** Place the Launch Ball button with its center 4-1/4" below the plunger's main drill center.



- **Plunger + Launch button, button on top:** Place the Launch Ball button with its center 2-1/2" above the plunger's main drill center.



Reference: Chapter 21, Cab Body - Plunger and Launch button

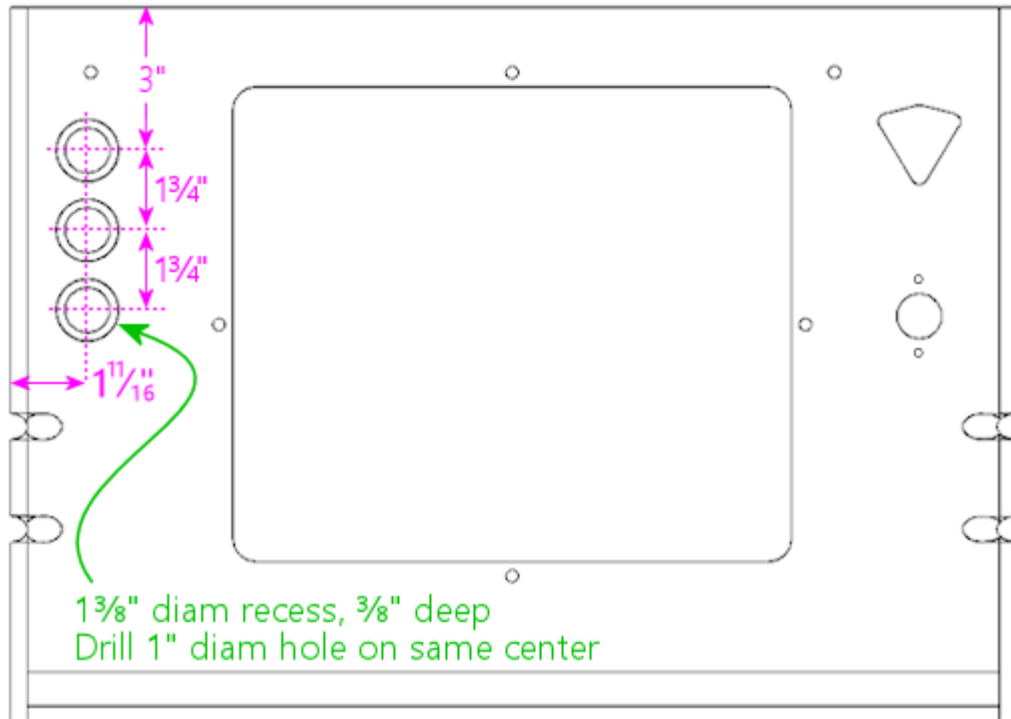
Drill the front panel buttons

For SuzoHapp small round pushbuttons (the standard part used for the Start button on most machines since the 1990s): Using a Forstner bit, drill a 1-3/8" diameter

inset to about 3/8" depth. Then drill the rest of the way through on the same center with a 1" diameter Forstner bit or hole saw.

Placement: For virtual cabs, you can put as many buttons as you like wherever you like. But the available space limits the options, and most people end up putting one to three buttons to the left of the coin door. A three-button layout that fits a standard cab (and fits with the standard hardware) is shown below. If you use your own layout, make sure that it doesn't conflict with the lockbar, leg brackets, or coin door.

For a replacement cab for a real pinball machine, it's best to measure a factory original and replicate its layout. The buttons might need to align with other cabinet hardware specific to the title, and/or with the cabinet artwork.



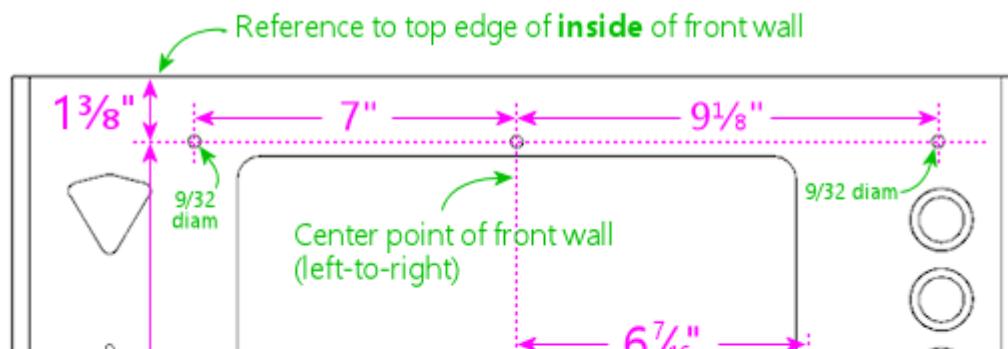
Reference: Chapter 21, Cab Body - Front panel buttons

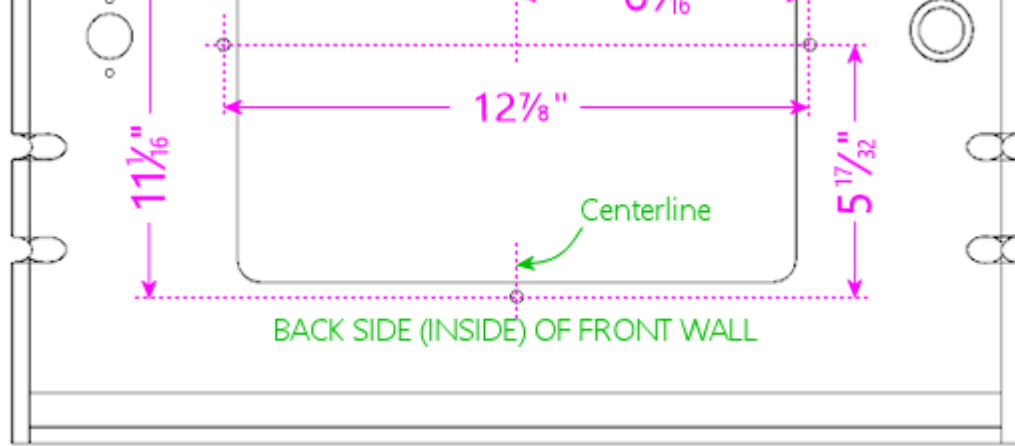
Drill the lockbar receiver and coin door bolts

Note: see Test the lockbar fit above before proceeding.

Drill the six 9/32" holes shown in the diagram (three across the top for the lockbar receiver, and three more around the perimeter of the coin door).

Reference the vertical position to the **inside** top edge of the front panel. Center the middle bolts horizontally in the panel (this should also be the center of the coin door cutout).





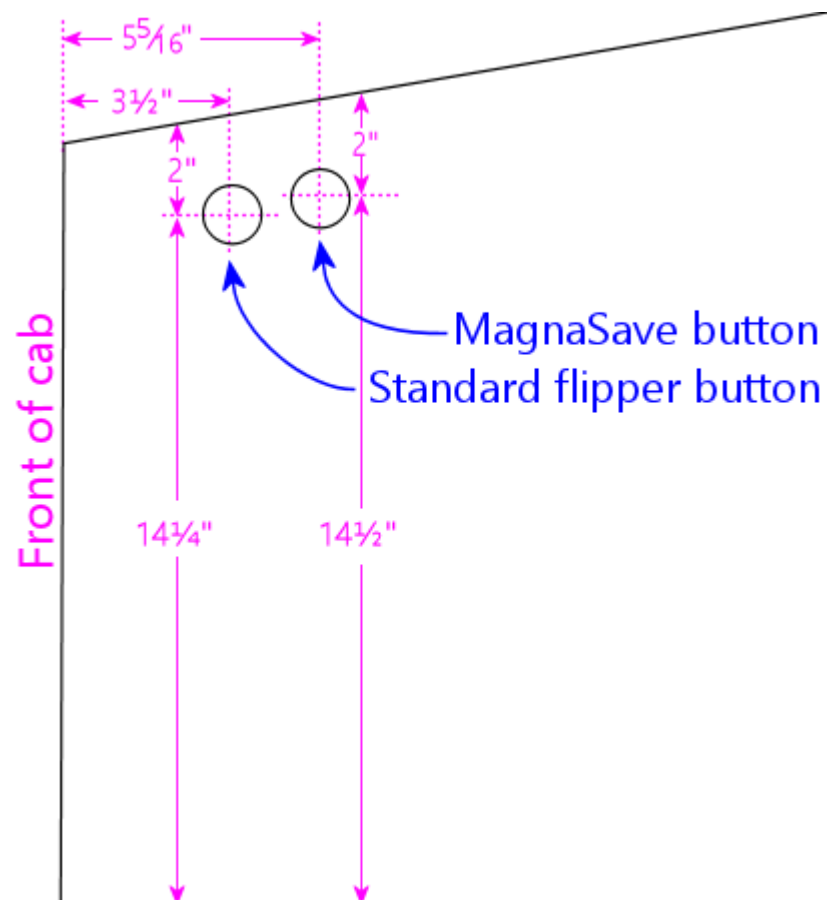
References:

- Chapter 21, Cab Body - Lockbar receiver
- Chapter 21, Cab Body - Coin door cutout

Drill the flipper buttons

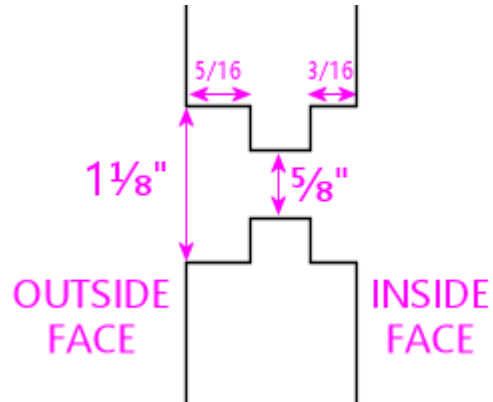
For WPC-style side rails (narrow rails that don't cover the flipper buttons), drill at the locations shown below. The flipper button goes at the same location whether or not you're including a MagnaSave button.

For System 11 rails or other wide rails that cover the flipper buttons, don't use these coordinates. Instead, drill at the same location as the pre-cut button hole in the rail, using the rail itself as a template.



How to drill:

- Original pattern used in most commercial machines (see diagram below):
 - Drill a small pilot hole (1/8") on the center, all the way through; use this as the center for all of the remaining drills
 - Use a 1 1/8" Forstner bit or hole saw to drill a 5/16"-deep depression from the **outside**
 - Use the same 1 1/8" bit to drill a 3/16"-deep depression from the **inside**
 - Drill the rest of the way through with a 5/8" bit



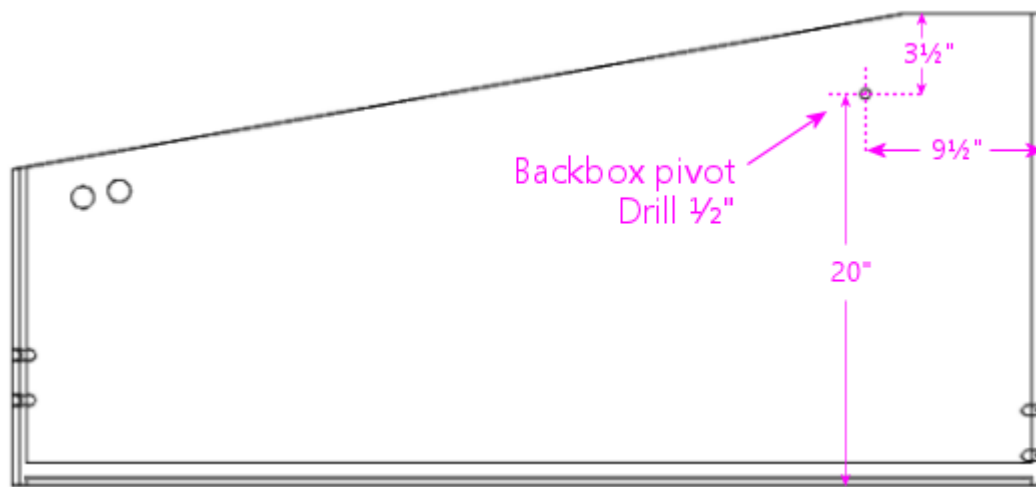
- Simplified alternative: If you're using an LED board or VirtuaPin flipper switch bracket, drill straight through with a 1-1/8" diameter hole saw or Forstner bit.

Reference: Chapter 21, Cab Body - Flipper buttons

Drill the backbox hinge pivots

Note: You might want to wait until after cabinet assembly to drill these holes, to fine-tune the positions based on aligning the backbox perfectly in the final fit. See the **Alternative Procedure** under Post assembly: drill the backbox hinge bolts below.

If you want to pre-drill these holes, drill a 1/2" diameter hole in each side of the cabinet, 9-1/2" from the back edge and 20" from the bottom edge.



Reference: Chapter 21, Cab Body - Side walls

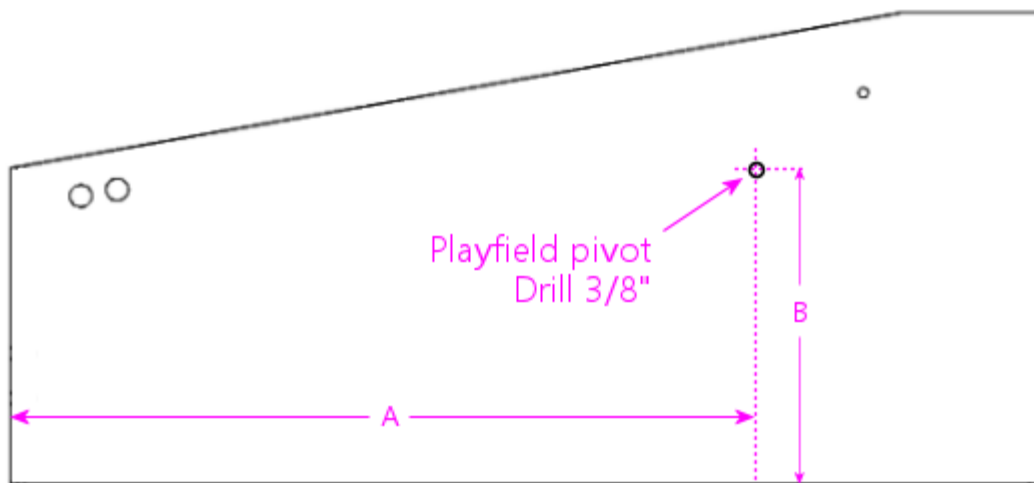
Drill the playfield pivots

This step applies only to replacement cabinets for Williams/Bally System 11 and WPC titles. Don't drill these holes for a virtual cab.

Single pivot nut system (games through mid 1992): The playfield is supported on a single pivot point on each side wall. The pivot point in older games (before about 1990) is a steel bushing (essentially a cylindrical steel spacer), 3/8" inside diameter, 1/2" outside diameter, that fits over a 3/8"-16 carriage bolt attached from the outside of the cabinet. The bushing was superseded by a 3/8"-16 threaded pivot nut starting in about 1990, and you can replace the bushings on older games with the pivot nuts when refurbishing, if desired.

The location of the pivots varies by title. I've collected measurements for a few machines listed below. If your specific machine isn't in the table below, you'll have to find a factory original to take measurements from. I'd like to expand this into a more comprehensive list, so if you have trustworthy information for a machine with a WPC-style cab that's not listed below, please send it to me.

Be especially careful with these to drill the holes perfectly straight and to line them up as precisely as possible on the two sides. The playfield won't seat properly if the pivot nuts are angled or misaligned.

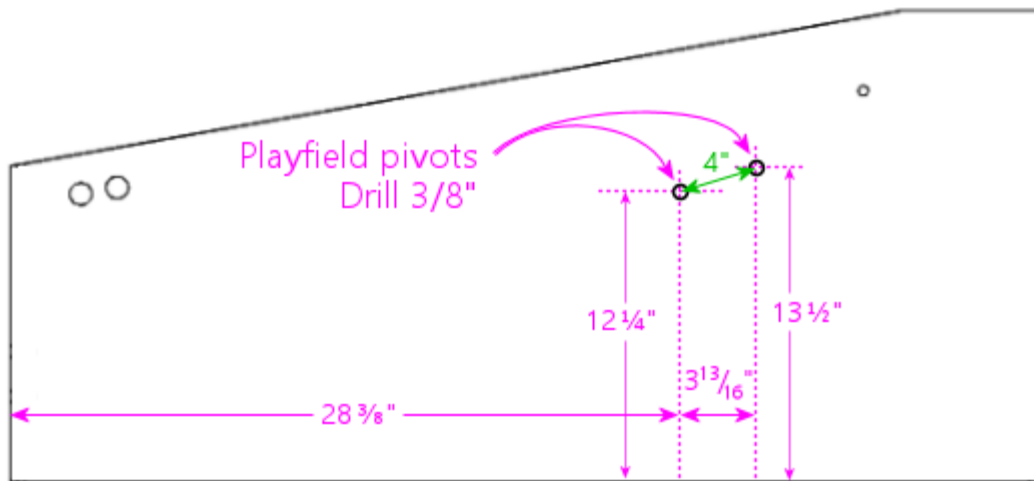


*Playfield pivot for machines with single pivot nut. Drill 3/8". The drills are the same on the left and right sides. The location varies by title - see table. **A** = distance from front corner, **B** = distance from bottom edge. Note that **A** is the distance to the front outside corner of the **finished cabinet**, so if you're measuring it before assembly, adjust for joinery. No joinery adjustment is needed for mitered joins, since the outer face of the panel extends all the way to the finished corner.*

Title	A	B	Pivot Nut	Carriage Bolt
<i>The Addams Family</i>	36-3/4"	14-7/16"	#02-4329 (1/2")	4322-01123-20B (1-1/4")
<i>Whirlwind</i>	36-3/4"	14-7/16"	#02-4324 (See note)	4322-01123-20B (1-1/4")

Note: Part 02-4324 is no longer available from any of the pinball vendors; it was a steel bushing/spacer with 3/8" ID, 1/2" OD, length 1/2", typically secured with a hex nut. One of the threaded pivot nuts (02-4329 or 02-4329-1) should work as a substitute.

Slider bracket system (games from mid 1992): This applies to titles starting with *Fish Tales*. These support the playfield on the newer slider brackets (parts A-16637-1/A-16637-2 or A-17749.1-1/A-17749.1-2), which rest on two 3/8"-16 x 7/8" pivot nuts (02-4329-1) on each side. The pivot nuts fit over a spacer plate (01-11408) and mate with 3/8"-16 x 1-1/4" carriage bolts (4322-01123-20B).



*Playfield pivot for machines with single pivot nut. Drill two holes at 3/8". The drills are the same on the left and right sides. Note that the horizontal locations are measured from the front outside corner of the **finished cabinet**, so if you're measuring it before assembly, adjust for joinery. No joinery adjustment is needed for mitered joins, since the outer face of the panel extends all the way to the finished corner.*

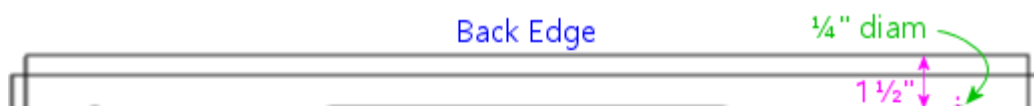
Note: I think the pivot nut locations are the same for all titles using the sliders, but I've only verified it against a couple of machines (*Theatre of Magic* and *Medieval Madness*). Swinks's plans on Pinside make the same assumption, although they differ from mine on the exact locations of the drills by 1/8" to 1/4". That might be due to measurement error, manufacturing variations, or actual design differences in the titles we sampled. (The titles mentioned in the Swinks thread are *Bram Stoker's Dracula*, *Creature from the Black Lagoon*, and Stern's *Iron Man*.) I think the slider system can tolerate this much variation without any functional impact, but even so, I'd measure a factory original of your particular title before drilling to make sure it really is in this range. If you know of a title that's significantly different from my figures, please let me know so I can include it in this section.

Drill the backbox floor hinge bolts

Note: I prefer to do this **after cabinet assembly**, by installing the hinges first, then getting the backbox aligned perfectly, and marking the bolt locations once everything's in position. This ensures a perfect final fit. See Post assembly: drill the backbox hinge bolts below.

Alternatively, you can do the same alignment the other way around: pre-drill the backbox floor hinge bolts, and defer drilling the pivot holes in the main cabinet until after doing a test fit.

If you want to pre-drill these holes, drill three 1/4" diameter holes, 1-3/4" apart, starting at 1-1/2" from the back edge, and 2-1/4" from the outside left/right edges.

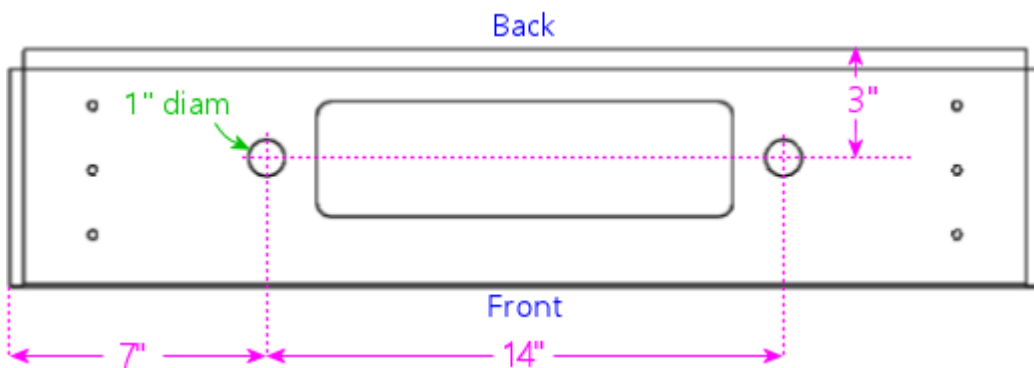




Reference: Chapter 21, Cab Body - Backbox floor

Drill the backbox floor wing screws

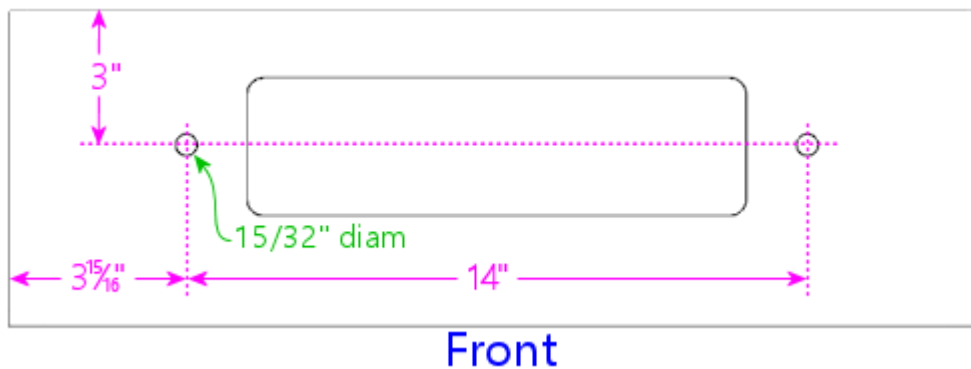
Drill two 1" diameter holes in the backbox floor as shown below.



Reference: Chapter 21, Cab Body - Backbox floor

Drill the shelf wing screw T-nuts

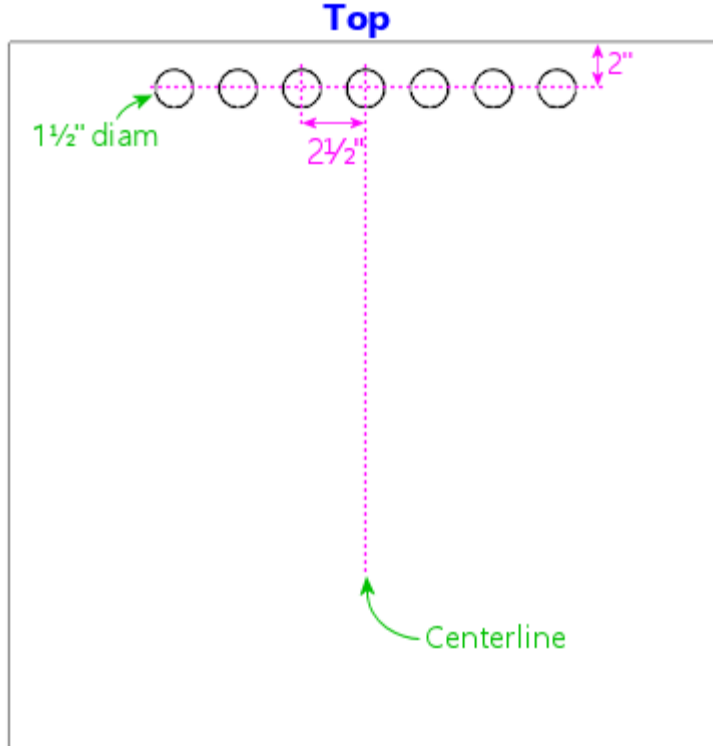
Drill two holes in the rear shelf, sized for 3/8"-16 tee nuts (typically 15/32" diameter), as shown below. Note: these should line up (on the same centers) with the backbox floor wing screw holes when the rear edges of the two pieces are flush.



Reference: Chapter 21, Cab Body - Rear shelf

Drill the backbox vents

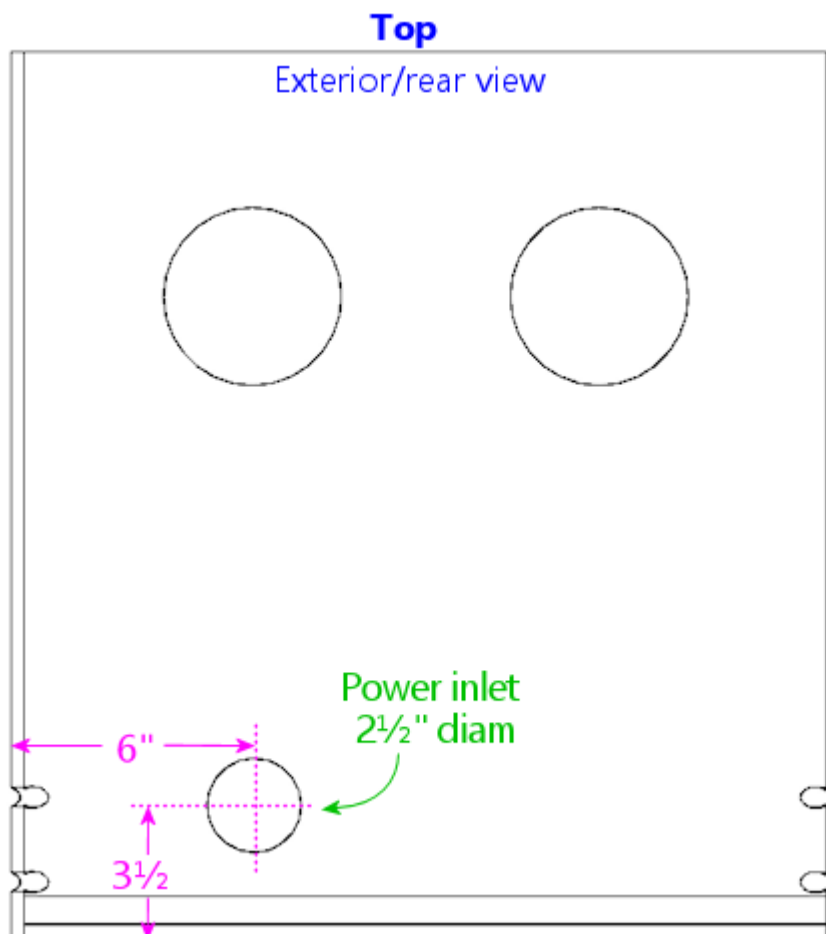
Drill 7 holes in the back wall of the backbox, 1-1/2" diameter, with the centers 2" from the top edge and spaced 2-1/2" from center to center, as shown below.



Reference: Chapter 21, Cab Body - Backbox back wall

Drill the back wall power inlet

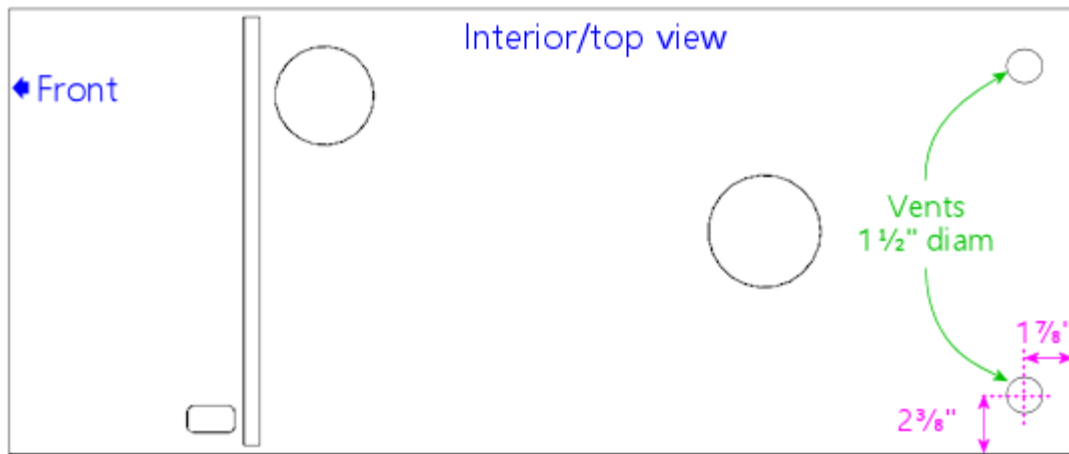
Drill a 2-1/2" diameter hole in the back wall, as shown below.



Reference: Chapter 21, Cab Body - Rear wall

Drill the floor vents

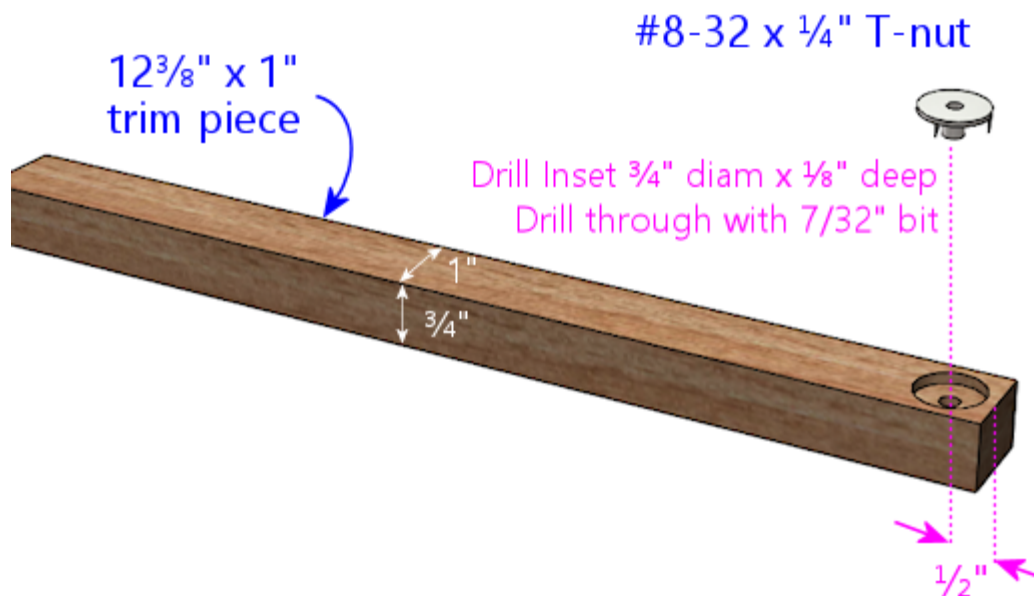
Drill two 1-1/2" diameter holes near the rear of the main cabinet floor, as shown below. (These are passive air vents for cooling on the original WPC cabs. You don't really need these on a virtual cab if you already cut separate floor openings for intake fans.)



Reference: Chapter 21, Cab Body - Floor

Drill the translite lock T-nuts

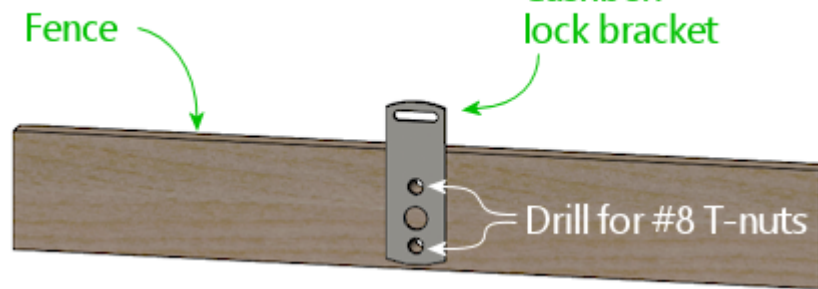
If you're installing a translite lock in the backbox, drill holes for tee nuts in the 12-3/8" x 1" trim pieces as shown below.



Reference: Chapter 21, Cab Body - Translite lock plate preparation

Drill the cashbox fence T-nuts

If you're installing a cashbox fence and cashbox lock bracket, drill holes in the fence for the tee nuts. Use the bracket as a drilling template, and drill for #8 tee nuts (typically 7/32" to 1/4").

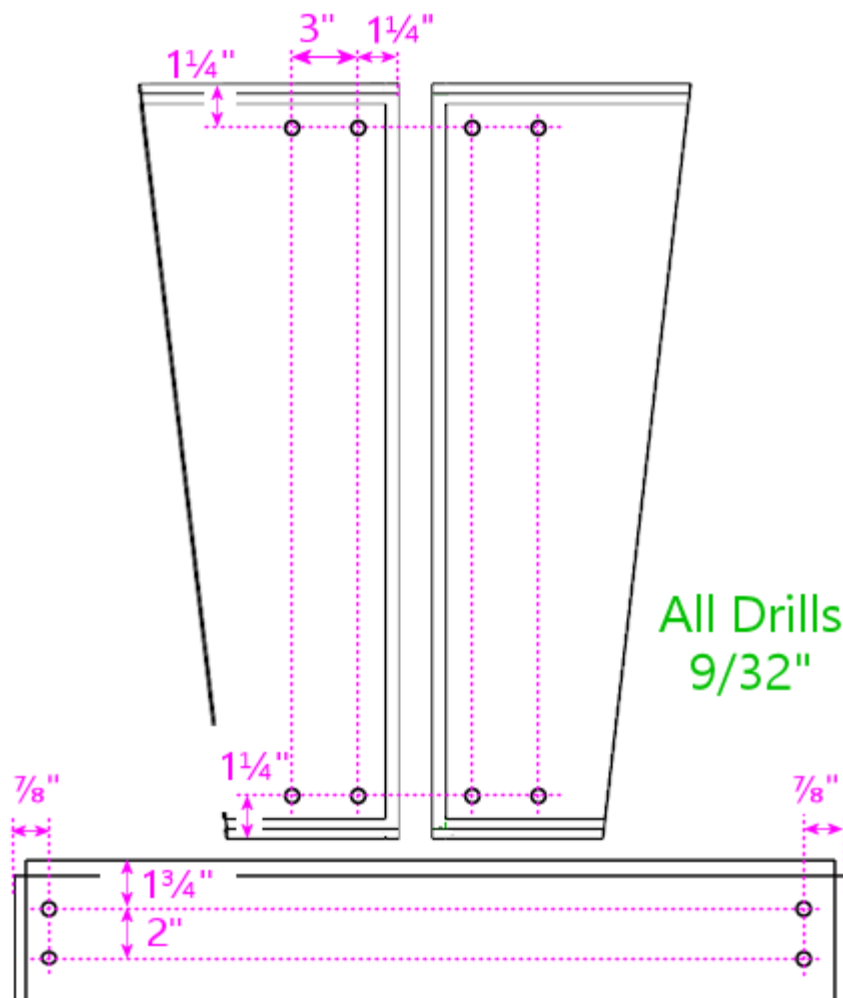


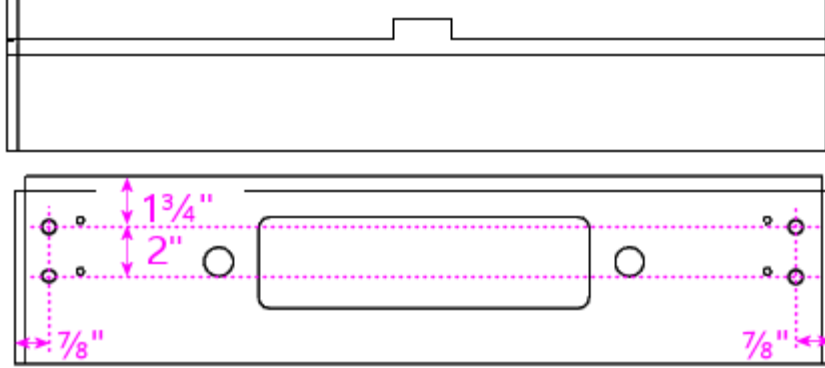
Reference: Chapter 21, Cab Body - Cashbox fence

Drill the backbox corner brace bolts

This step only applies if you're using corner braces in the backbox, Williams part #01-9167, which attach with 1/4"-20 carriage bolts. Drill four 9/32" holes in each of the backbox top, bottom, left side, and right side panels, as shown below.

Note for widebody/custom widths: the #01-9167 corner braces can't be used on the bottom corners with a widebody cabinet, because the width of the cabinet requires the hinge brackets to be placed further apart, bringing them into conflict with the corner brackets. There's a special version of the corner bracket for widebody cabinets that fits over the hinge bracket and doesn't require any additional holes in the bottom panel. VirtuaPin sells the wide brackets under part number 01-9167-W. (I'm not sure if that's the official part number - I can't find it listed in any of the Williams parts manuals or for sale from any of the other pinball vendors.) For custom-width cabinets that are in between the standard and widebody sizes, you're likely to have the same conflict, with no easy way to resolve it, so I'd just skip the bottom corner braces.

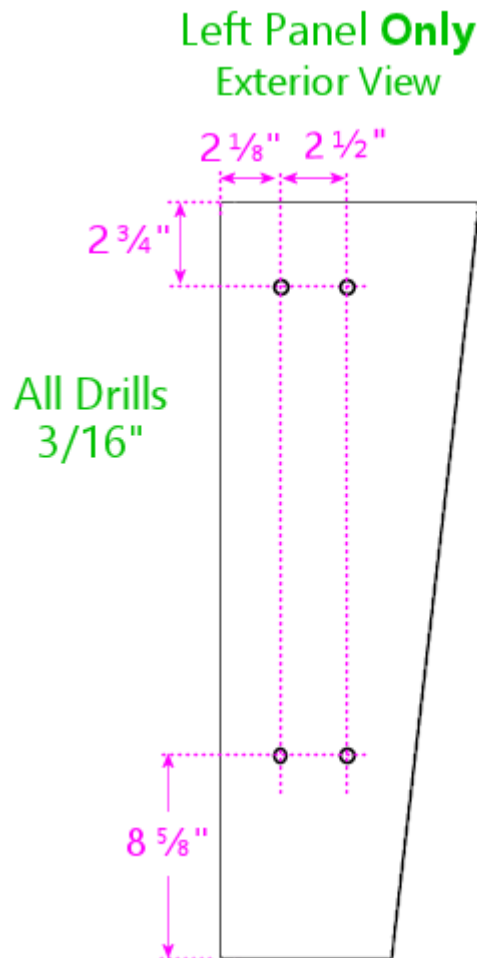




Reference: Chapter 21, Cab Body - Backbox corner bracing

Drill the backbox insert panel bracket bolts

This applies only to replacement cabinets for real pinballs, for WPC and System 11 games that use a backbox insert (the plywood panel with lamps that sits behind the backglass to provide back-lighting). **Don't** drill these holes for virtual cabinets or for later WPC games with plastic "tub" inserts. These drills are for #10 carriage bolts, which fasten the insert hinge brackets, parts A-12497 (upper) and A-12498 (lower), to the inside of the left wall of the backbox. Drill four 3/16" holes as shown, in the **left side panel only**.



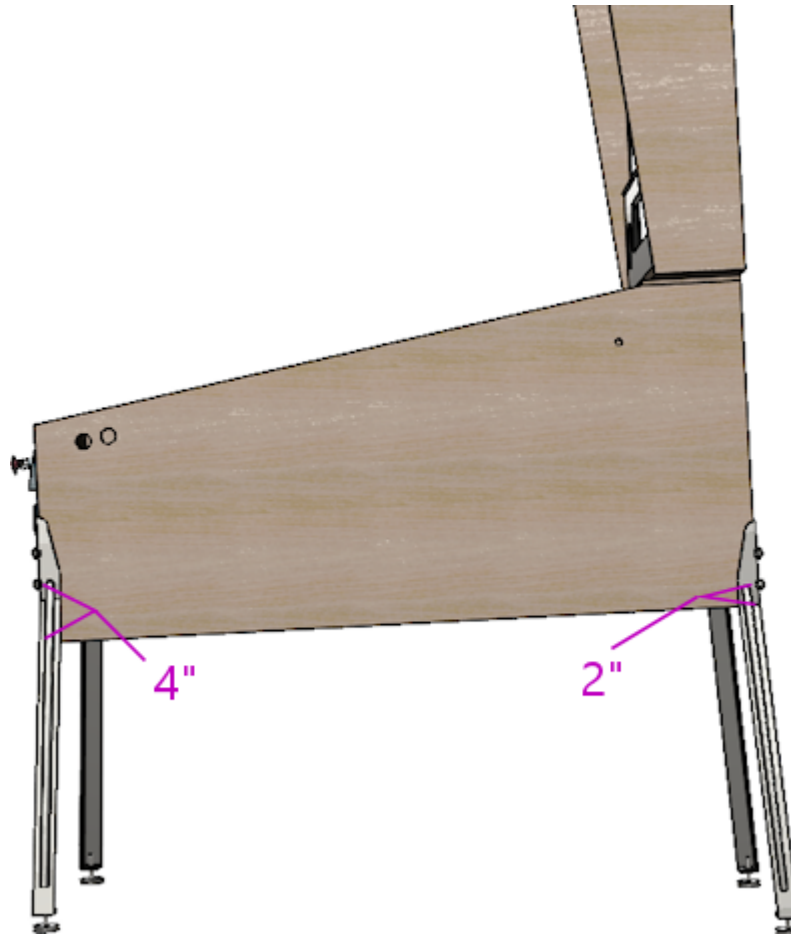
Note: These drill locations are the same on several machines I've checked, from a mix of System 11 and WPC-95 titles, but I haven't done an exhaustive survey. If possible, verify the measurements for your specific title by checking against a factory original.

Post-assembly: drill the leg bolts

Two 3/8" drills in each corner of the main cabinet, 2-1/4" apart, at a 45° angle into the corner.

Front left/right: drill centers at 4" and 6-1/4" from the bottom edge

Back left/right: drill centers at 2" and 4-1/4" from the bottom edge



Reference: Chapter 21, Cab Body - Leg bolts

Post-assembly: drill the backbox hinge bolts

Attach the backbox hinges to the main cabinet with their pivot nuts.

Set the backbox on top of the cabinet. Center the backbox left to right, and align the back of the backbox flush with the back of the cabinet. **Secure the backbox in this position** with wing screws, screwing them through the holes in the floor of the backbox and into the matching T-nuts under the shelf.

Rotate the hinges up so that they sit flat against the bottom of the backbox, and mark the locations of the three bolt holes. Do this for both sides. Before marking positions, make sure that the hinges are parallel to the sides of the cabinet, and make sure there's enough of a gap that they won't rub against the sides when the backbox is rotated.

Remove the backbox and drill 1/4" holes at the marked positions.

Alternative procedure: Drill the six bolt holes in the backbox floor first, but **do not** drill the pivot bolt holes in the cabinet sides yet. Attach the cabinet hinges to the backbox with six carriage bolts (1/4"-20 x 1-1/4") and whiz flange nuts (1/4"-20).

Position the backbox on top of the main cabinet as described above and secure it with wing screws. Mark the hinge pivot hole positions on the sides of the cabinet. Remove the backbox. Drill a 1/2" diameter hole on each side of the cabinet at the marked position.

References:

- Chapter 21, Cab Body - Backbox floor - Hinge bolts
- Chapter 24, Backbox Hardware - Backbox hinges

Appendix 11. A Few Woodworking Tips

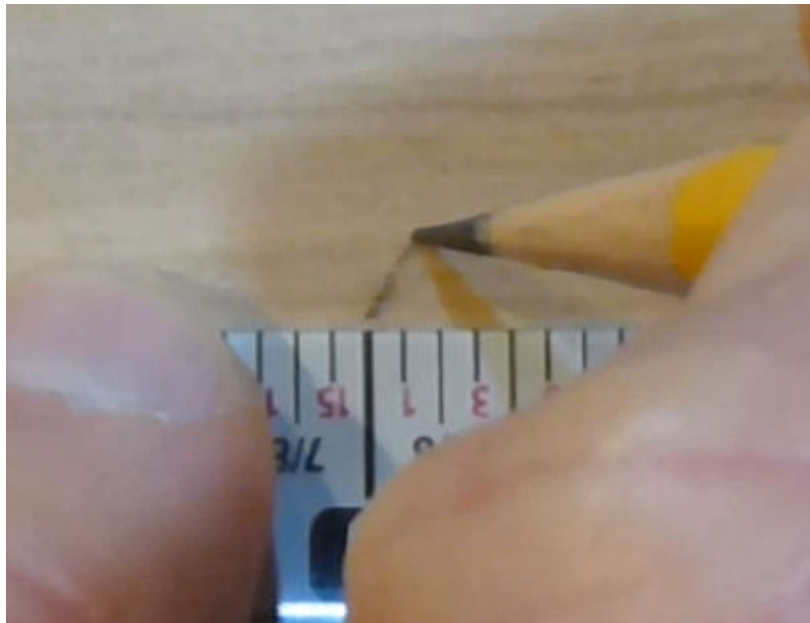
I'm such a woodworking newbie that I'm a little shy about including a section on "tips". But I'm doing it anyway, because I've come across a few useful things that don't seem to be widely advertised, and I want to pass them along. This isn't even remotely a tutorial on woodworking - it's just a few miscellaneous ideas that I've found helpful.

I picked up most of the tips here from Web tutorials and Youtube videos. You can easily find these tips and much more with a little online research. But the Web is vast, so sometimes it's nice to have a curated list of highlights from someone who's not just trying to pad out a listicle.

Measuring and marking lines

If you're accustomed to marking measurements on a board with a little "tick" mark with a pencil, here's a slightly different method that's no more work, but really improves my accuracy.

- Put your pencil tip right against the mark on the tape or ruler at the point you want to mark
- Make a diagonal stroke out to the right



- Put your pencil tip back at the same tape/ruler mark
- Make a second diagonal stroke out to the left





Now you have a little "V", with the point of the "V" exactly at the measured location.



The repeatability of my measurements improved noticeably when I starting using this technique. The old tick mark method leaves a surprising amount of fuzziness, mostly because the little line is never perfectly straight (not when I draw it, anyway). The "V" marks a single point, instead of a line, so you can see exactly where the measurement was supposed to be.

Tape measures

The best measuring tool for many woodworking tasks is a tape measure. I used to think rulers were better when you needed an accurate measurement, but a tape measure is actually better for woodworking in many cases, if you use it right. The key is the little hook on the end.

When you want to measure from the edge of a board, hook the tape onto the edge you want to measure from, and pull the tape tight across the board to find the point to mark. The hook sets the zero point exactly at the edge of the board. This is much more precise than trying to align a ruler with the edge.

Here are two tips for getting better accuracy out of your tape measure:

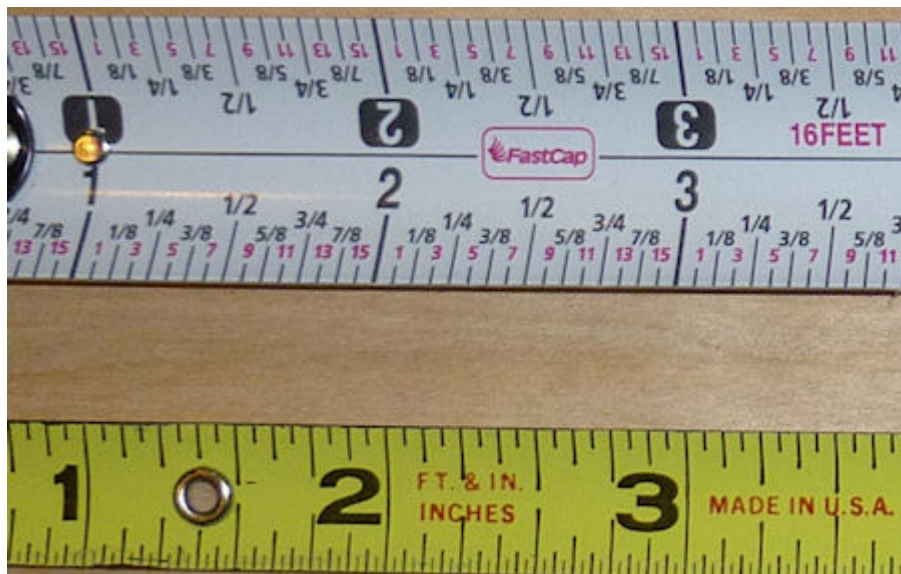
- The little hook on the end of the tape is **supposed to be** loose like that. I always thought the looseness in my old tape measure was from age - I thought the little rivets holding the hook on had loosened up over time. But the sliding hook is actually a feature, not a bug. It's there to precisely account for the thickness of the hook itself. When you attach the hook to the outside edge of a board, pull it tight, and the hook will align so that your reading on the tape corresponds to the distance from the edge of the board. When you're measuring the inside of a space, push the hook up against the wall you want to

measure from, so that the hook slides all the way inward, and now the reading on the tape is precisely the distance from the thing it's pressed up against.

- Use **the same tape measure** for all measurements that have to align with each other or be consistent with each other. Tape measures (and rulers as well) can have a fair amount of variation when you compare them side by side - it seems typical to see differences of 1/32" to 1/8" over distances of a few feet. Even a variation as large as 1/8" over 3' is better than 99.5% accuracy, which is great in most contexts, but those fractions of an inch can really matter in woodworking. Anyway, you can make those errors cancel out in most cases simply by using the same tape measure for all related measurements. That way, at least the readings will all be off by the same amount, so they'll line up with each other after cutting.

By the same token, if you need a measurement to line up with some pre-fab part (a metal trim piece, say), measure that part with the same tape measure rather than relying on the specs. Or just use the part itself as a ruler. Don't count on every part to match the specs perfectly; a lot of this stuff isn't manufactured to the highest precision.

Buying tip: I find it really helpful to have a tape where every tick mark on the tape is marked with a number. I have an older tape that only has numbers on the inch marks, and a newer tape with every 1/16th tick mark individually numbered. I've made more mistakes than I'd like to admit reading the old tape, where I misread a 1/16th tick because I wasn't paying close enough attention or I didn't have good enough lighting. It's harder to screw that up when the number is printed right on top of the mark you're reading.

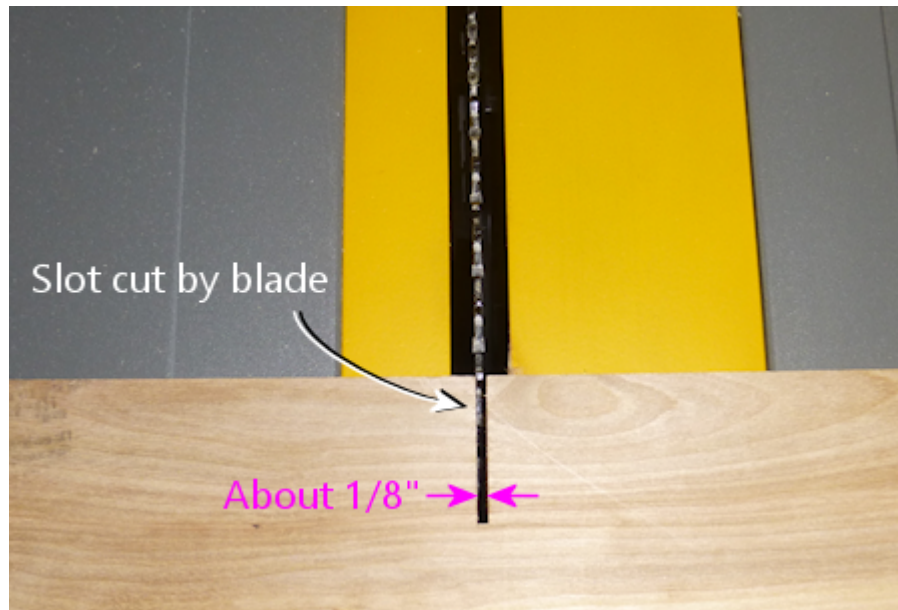


Comparison of tape measure markings. Every 1/16" mark is numbered on the top tape. On the bottom tape, only the inches are numbered, and you have to infer the fractions - which isn't exactly hard, but it's more error-prone. On the other hand, the bottom tape is narrower, which makes it easier to hold the edge flat against a board to precisely line up a mark. My ideal would be a narrow tape like the bottom one with every tick mark clearly numbered like the top one.

Sawing on the "right" side of the line

If you start with a board that's 12" wide, and you use a saw to cut it exactly in half, how wide are the two resulting pieces? Strangely, the answer isn't 6". It's actually more like 5-15/16". What happened to the missing 1/8"? It basically vanished into

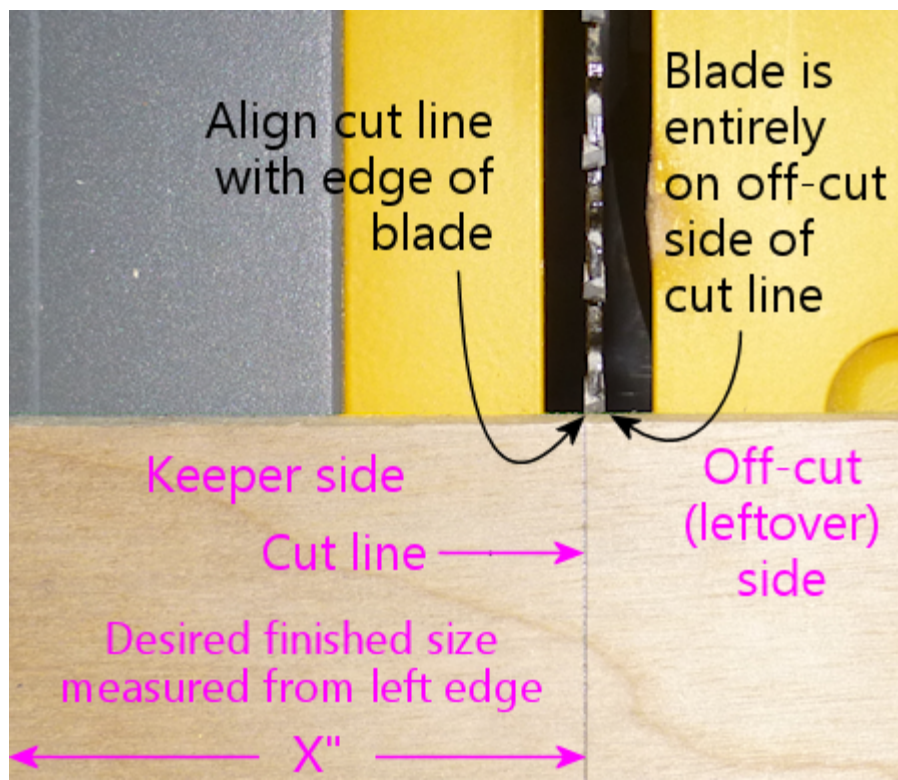
thin air. Well, almost: it got converted into sawdust. The technical term for this is the "kerf" of the blade - the width of the slit that the blade makes as it cuts through the board. You can easily see this in action by making a cut partway into a board.



Partial cut, showing the slit that the saw leaves as it cuts. Woodworkers call the width the "kerf" of the blade.

The usual goal when you're sawing a board is to produce a finished board that's an exact target width. You don't want it to be "about" the desired width, give or take a few sixteenths. You want the result to be just what you measured.

The way to get the finished board to match the measured size is to place the blade on the outside of the cut line. In other words, position the saw so that the entire width of the blade is within *leftover* part of the board. Align the inside face of the saw blade (the face closer to the keeper part of the board) with the cut line. That way, the slit that the saw cuts will be taken out of the leftover part, leaving exactly the measured size for the keeper part.



How you apply this varies a little bit with different types of saws:

- For a table saw, making a rip cut using the fence, the keeper part is the side that goes against the fence. The fence gauge should always be calibrated to the distance between the fence and the blade, so the reading on the gauge should reliably give you the width of the finished piece that comes out of the saw on the fence side.
- For a cross-cut with a table saw, using a sled or miter gauge (rather than the fence), mark the cut line on the board, and align the cut line against the saw blade so that the whole width of the blade is on the leftover side of the line.
- For a track saw, mark the cut line on the board, and position the track so that it's on top of the keeper side of the board, with the edge of the track right up against the cut line.

Drilling clean holes

The big problem that everyone encounters when drilling plywood is splintering on the back side of the board, where the bit exits.

This seems to be a vexing problem for a lot of people, given the bounty of How To pages about it on the Web. The usual advice is to always use brand new drill bits, or to use special bits like stepped bits or Forstner bits. It's hard to argue with the notion that a sharp bit will drill cleaner than a dull one, but it's hard to imagine anyone actually follows advice that you should treat your drill bits as single-use disposables. And the special bits have their uses, but none of them can fix the back-side splintering problem by themselves. The splintering happens because of the way plywood is constructed, not because of the type of drill bit you're using.

Another approach you often see recommended is to drill halfway through from each side, meeting in the middle. This avoids back-side splintering by essentially eliminating a back side. That can indeed fix the splintering problem, but I've found that it often creates a new problem, which is that it's difficult to get the two half-holes to align perfectly. There's usually at least a tiny offset between them, no matter how careful you are. You can pre-drill with a small pilot bit to mark the center point on both sides, but even then, the accuracy is limited to the non-zero size of the pilot hole. In some cases, a slight offset might not matter, but it can be problematic when precision is required, such as when you need a good fit for a bolt.

What actually works: There's only one technique I've found that really works, and the good news is that it works reliably, with almost any drill bit. It's also pretty easy. The technique that works is to **use a backing board**. In other words, take a piece of flat scrap material that you don't mind drilling an extra hole in, and clamp it tightly to the back of the board, behind where you're drilling the hole. Tight clamping is the key - it's what makes this work. Now just drill straight through from the top side.

The whole reason splintering happens in the first place is that the veneer tends to bend outwards just before the bit penetrates it. As the bit gets close to breaking through, it cuts away at the veneer from the inside, making it thinner and thinner. This reaches a point where the veneer is so thin and fragile that the pushing force of the bit overwhelms the cutting action, bending the super-thin fragment of veneer outwards instead of cutting it. The wood grain holds this bit of bending veneer together and pulls it up and away from the board for some distance away from the hole, resulting in those torn splinters around the exit point of the drill.

The backing board fixes this by providing an extra layer of strength outside the

veneer that holds the veneer flat until the bit is all the way through.

I almost always get clean holes on both sides of the board when using this technique, as long as the backing board is clamped tightly enough. This works for regular spiral bits, Forstner bits, and hole saws.

Cutting rectangular openings

It's easy to cut a rectangular opening with a jigsaw, but I've never been able to get professional looking results. My jigsaw cuts always end up at least a little crooked. I get much nicer results with a router and an edge guide, but it's extremely tedious to cut a large opening in a thick board this way, because you have to make multiple passes. You're only supposed to route about 1/4" of depth at a time. The router also produces a ridiculous amount of sawdust.

The best solution I've found for cutting large openings is to combine the jigsaw and router methods, in a two-step procedure:

1. Make a rough cut with a jigsaw, leaving a safety margin of perhaps 1/8" inside the bounds you want to cut
2. Finish each side with a router and a straight bit, using an edge guide to get a straight line at the final boundary

For the initial cut with the jigsaw, drill holes near the four corners big enough for the jigsaw blade. Make sure that the holes are inset from the final corners far enough that you don't drill any material outside the lines. Now use the jigsaw to make a rough cut along each edge, staying well within the final outline. The point is to maintain a safety margin so that if the jigsaw swerves off course, you'll be able to stop it before it cuts anything outside the final outline. Any jagged lines at this point don't matter because they're all inside the interior of the cutout, which is going to end up being entirely removed when we're done.

After the rough opening is finished, you can use the straight router bit to trim back each edge to the final cut line. You can set the router bit height to the full thickness of the board, since we don't have to make a "plunge" cut into the board. We're just going to nibble away at the edge until we reach the cut line. This is what makes this approach faster than doing the whole thing with the router bit: when using just the router, you have to make the cut in 1/4" depth increments, because router bits can burn the board if you try to cut deeper than that all at once. But this way, we're going in from the side instead of straight down, so we can do a little bit at a time sideways instead of a little bit at a time in depth.

For the routing steps, I work on one edge at a time. I clamp a straight edge to the work piece to serve as the guide, parallel to the cut line, offset by the distance between the router bit and the outside of the router base plate. The guide can simply be a straight piece of plywood, or any other good straight edge, such an aluminum level. You can measure the distance on the router itself, but I find it works much better to make a test cut against a straight edge and measure the distance between the straight edge and the cut. Then you just position the straight edge the exact same distance from the line you want to cut. Clamp it to the board. Position the router bit in the jigsaw opening, making sure it's in a completely open area, not in contact with any wood at this point. Turn on the router and let it come up to speed. Gently and gradually move the bit into the target edge, until the base reaches the straight edge. Slowly move the router along the straight edge. Stop a little short of the corners, to make sure you don't put a dent into the corner that goes outside the box. Repeat for each edge.

By the way, this technique also works for other shapes with straight edges, such as the triangular(-ish) cutout for the plunger in the front wall of a pin cab.

There's a right direction when routing

Routers have a preferred direction for moving them across a work piece. I didn't really appreciate what this meant even after reading about it in my router's instruction manual, but it's actually kind of important.

With most routers, you're supposed to move the router clockwise when you're routing the inside of an opening, and move it counter-clockwise when routing outside a perimeter. (Check your router's manual to make sure it agrees, but I think it's universal - it's a function of the bit spin direction, which I think is standardized across all manufacturers.)

The preferred direction is important because it stabilizes the router's motion, helping you keep the router under control. When the bit contacts the work piece, it applies a force to the whole router. Moving in the preferred direction helps direct this force for good rather than evil - it tends to push the router towards the work piece, which keeps the motion stable and under your control. If you move the router in the backwards direction, the contact force can make the router kick and jerk away from the work piece, often with enough force to momentarily overwhelm your steering guidance. Apart from the obvious danger of injury, this can easily ruin the workpiece by jerking the router away from the intended cut line.

I keep a cheat-sheet with a diagram of the rule on my workbench, so that it's in view whenever I use the router.

Appendix 12. Lock Miter I: The Plywood-Friendly Way

The original Williams pinball cabinets of the 1990s used a type of corner join known as a lock miter. This join has mitered corners (meaning the ends of the pieces that come together at a corner are cut at 45°, so that the seam is exactly the corner, making it practically invisible), plus an interlocking tongue-and-groove pattern on the inside (that's the "lock" part). There are other good ways to make the corner joins, but the lock miter is one of the best. The corners are seamless, the joints are self-aligning and self-squaring, and they have a lot of internal surface area for strong glue adhesion. Lock miter corners are also make assembly really easy, since the corners snap together like jigsaw puzzle pieces.

There are (at least) two ways to make a lock miter join. The first method, which we'll cover in this chapter, involves making a sequence of cuts (six steps in all) with a 1/4" straight router bit and a table saw. I'm covering this method first because it works very well with plywood, which is what most pin cabs are made of. This method might look a little intimidating at first, because of the number of steps involved, but it's actually not that bad. Most of the steps are pretty simple, and they mostly use natural alignment points that don't require any arithmetic or ruler measurements.

The second method involves a special router bit that cuts the entire tongue-and-groove pattern all at once. That's covered in the next chapter, Appendix 13, Lock Miter II: The Special Router Bit Way. That method probably looks easier at first glance, since the bit does all of the work for you. But these bits are notoriously difficult to set up properly. There is a reliable setup procedure, but the time and effort involved in that ends up making the overall degree of difficulty similar to the more manual "plywood friendly" method. The real drawback of the special-router-bit approach is that the tongue-and-groove structures that the router bits cut are too fine for most plywood, so you lose a lot of the strength that the join is supposed to provide.

If you're new to woodworking, both methods are probably going to look like "advanced" procedures that should only be tackled by grizzled veterans with lavishly equipped wood shops. But don't give up yet! I'm a woodworking newbie myself, and I've found that both techniques are doable and can yield great results, even if you're new to this. I've tried to reduce both methods to straightforward recipes that you can follow without any special woodworking expertise and with a modest set of tools.

Example of the manual method

Here are some photos of a sample corner built with the manual method, to give you a better idea of what it looks like.





Dry fit of a test corner built using the manual lock miter method. This is just a couple of scrap pieces that I ran through the process in this chapter as a beta test. It took about an hour of work; most of that was spent on the "sneaking up" phase for the 45° angled cuts. It came out just about perfect - the two pieces fit together nicely, and the corner is razor-sharp.



A closeup of the grooves, with the front/back piece on the left and the side piece on the right.





Another view of the grooves. The thing I like about this version of the lock miter (the "manual" approach) is that the tongue-and-groove structure comes out so clean and sturdy when rendered in plywood, thanks to the relatively large features. Compare this to the similar pictures of the special router bit results in Appendix 13, Lock Miter II: The Special Router Bit Way).

Pros and cons of the two methods

If you're considering using lock miter corners for your pin cab project, I'd suggest reading through both methods, so that you can judge for yourself which one you feel more comfortable with. Here's a quick summary of what I think are the high and low points of each approach.

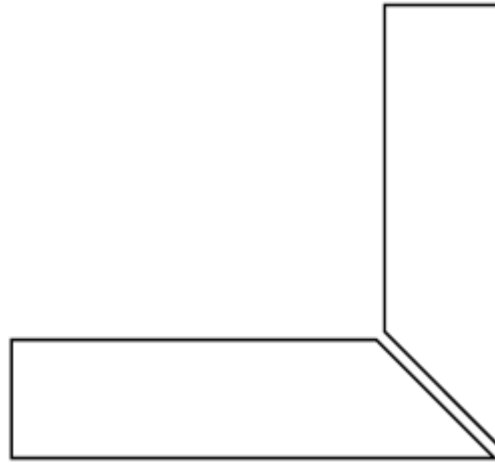
Router Bit Method	Manual Method
Special router bit required (\$25-\$100+)	Needs only a 1/4" straight router bit (and can be done with just a table saw, no router needed)
No table saw required	Requires a table saw
Not great for plywood; tongue-and-groove features have poor structural integrity due to small size	Excellent results with plywood, makes a super strong joint
Only needs one routing pass over each edge	Six cutting steps required
Bit setup is notoriously difficult (but doable if you follow the right procedure)	Fairly straightforward setup for each cut (but there are still six of them)

I've tried both approaches, and I've found that - surprisingly - the manual method isn't really any harder or more time consuming than the router bit method. It might even be a little easier. You'd think that the special router bits would make things a whole lot easier, given that they cut the entire pattern in one pass over each edge, but the complex setup procedure makes up for it. The manual method involves multiple cuts for each corner, but each one is fairly quick and easy to set up, and the recipe is pretty reliable at getting all of the cuts to line up accurately with one another.

Given that neither method has a clear advantage in terms of difficulty level, my feeling is that the manual method has an edge for a pin cab project, simply because of its better performance with plywood. The main obstacle with the manual method for many people will probably be that it requires a table saw. If you don't have a table saw and you don't want to buy or rent one, the special-router-bit method is a good fallback; even though it's not ideal for plywood, it is workable with at least some plywood.

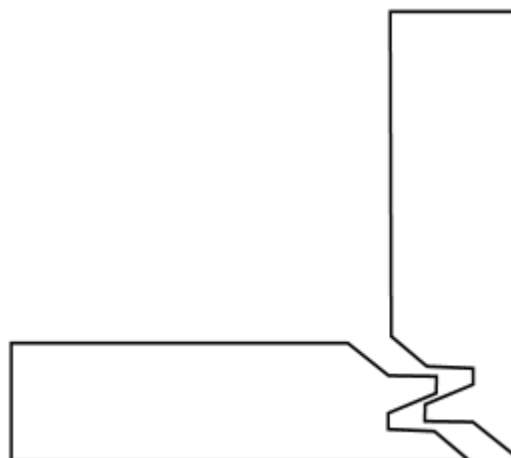
How the lock miter works

The lock miter is an elaboration of the plain miter join, which is a simple join where you cut the ends of the two pieces meeting at a corner at a 45° angle:

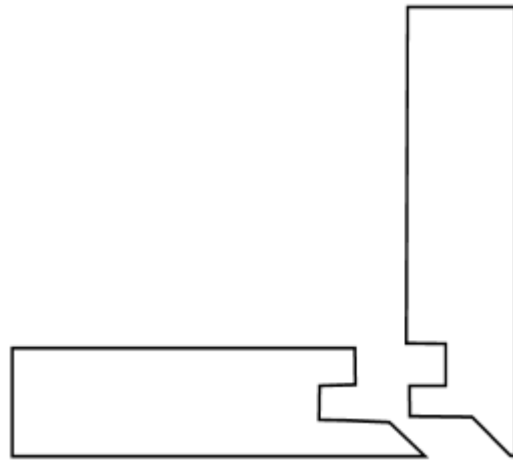


The plain miter gives you a nice clean corner without a visible seam, but it doesn't make a very strong glue joint, and it takes some skill to execute it well. For one thing, you have to get the two cuts *exactly* at 45°, or the corner won't be square. The way the two pieces meet means that any error in the cut angle of each piece is doubled in the assembled corner. If you're off by 1° on the cut angle, the corner will be out of square by 2°. The other thing that's difficult with a plain miter cut is getting the two pieces to align correctly during assembly. You have to get the corner aligned across its whole length, and then keep it there while the glue dries. There are corner clamp systems that help with this, but it still takes some skill.

The lock miter improves on this by keeping the 45° angle at the outside corner and adding an interlocking tongue and groove. There are different ways to make the interlocking tabs. The special lock miter router bits produce a pattern that looks roughly like this:



The manual method presented in this section produces this pattern:



The common features are the 45° angled cut at the outside corner, and the interlocking tongue and groove. Like the plain miter, the 45° angle at the outside makes the corner seamless. The addition of the locking tabs makes the assembly step almost effortless, since the pieces can only fit together at exactly the right alignment. And the extra surface area of the tabs forms a strong glue joint. This join is also a little more forgiving of imperfect 45° angles at the corner, since the locking tabs will still keep the corner aligned and square.

You probably noticed that the router bit method and the manual method have slightly different shapes for the interlocking tabs. The differences aren't completely arbitrary. The key difference is that the tongue-and-groove structures in the manual method are coarser. That's what makes the manual method perform better with plywood. The lock miter router bits are forced to use smaller features because their cutting pattern has to form a mirror image. That's how they work their magic trick of cutting both sides of a corner join with the same bit. Every feature has to appear twice, once on each side of the "mirror", so every feature has to be half-sized. The small projections are problematic with plywood, in that the glued plies tend to fall apart when cut so finely. The manual method doesn't have to be symmetrical, since we make each cut individually. That allows for larger features, which hold up better with plywood.

Equipment needed for the manual method

1/4" straight router bit: Just to be clear, we're talking about a bit with a 1/4" cutting diameter (as opposed the shank size - for that, simply match the shank size your router uses).

Router and router table: If you already have a hand router, you can buy an inexpensive bench-top router table and use it with your hand router. Most of the bench-top tables are compatible with many brands of routers, so you can generally mix and match tables and routers. I use a bench-top table from Skil with a Ryobi fixed-base hand router. The table isn't a high-end piece of precision equipment, but it works for the pin cab joinery I've attempted.

Router table fence micro-adjuster: Optional but really helpful. Provides a way to adjust the fence position in tiny fractions of an inch, to help get the alignment perfect. This is something you can make yourself as a simple DIY project described later in this chapter.

Table saw: For the 45° angled miter cuts.

1/4" MDF scraps: For use as alignment aids. You just need a couple of scrap pieces, roughly 6"x6" and 2"x6". 1/4" plywood or hard-board will also work.

Auxiliary fence for your table saw: A piece of 3/4" MDF or plywood, cut to about the same size as your table saw's fence.

Doing it all with a table saw

You can create a lock miter using just a table saw, no router required. You need a 1/4" dado stack in place of the 1/4" router bit used in my procedure.

The steps in my procedure are tailored to the router table, so you might want to look for a recipe that's specific to the table saw - there are some equally complete table-saw versions out there. Search the Web and/or Youtube for **lock miter with table saw**. Or you could just adapt my recipe, if you don't mind puzzling out how to translate the router fence alignment steps to the table saw. The alignment points for the cuts are all the same relative to the boards, so it doesn't take much translation. The main thing to watch out for is that you'll need an auxiliary fence for a couple of steps, since the saw blade will otherwise be too close to the main fence.

Do a practice run

This is a complex enough procedure that I felt the need to do a couple of practice runs on scrap material before I tried it for a real project. You might not need to, but I found it helpful as a confidence booster, to make sure I understood all of the steps.

The other nice thing about a practice run is that it helped me fine-tune my sense of the various alignment points. The procedure is designed to make the alignment points easy to judge, but even so, there's always a little wiggle room when trying to get two things to line up. If anything comes out a little off in the test run, you can use that as a guide to compensate the next time through.

Preparation

Get all of the panels ready: Wait until all of your panels are cut to final sizes before starting the lock miter procedure. You should do all of the lock miter routing across all panels at the same time, so that you only have to do the setup part of each step once.

I'd do the lock miter cutting on all panels **before** routing other features (e.g., floor dados), drilling holes (flipper buttons, say), and cutting openings (like the coin door cutout). The panels are easier to handle whole.

Include a scrap "side" and "front" piece for testing every cut. Reserve a small scrap piece of plywood for each, designating one as "side" and one as "front". Make each cut on the corresponding scrap piece first as a sanity check that you're set up correctly. This is especially helpful in the steps where you have to "sneak up" on the final alignments with a series of test cuts. The idea with "sneaking up" is that you start with a cut that's intentionally shy of the real cut line, then you check how close you got, and then gradually re-cut closer and closer until you get exactly to the right spot. I sometimes make the mistake of overshooting the final cut line because I wasn't gradual enough in the re-do cuts. It's nice to know that you're only ruining a piece of scrap if that happens.

Pre-mark all panels. The lock miter we're doing involves *two different cutting patterns*, one for the side panels, and a different one for the front/back panels. It's

crucially important that you cut the right pattern in each panel. To avoid switching things around in the heat of battle, mark each piece conspicuously before you start, to indicate whether it's a **SIDE** or **FRONT/BACK** piece. I also like to mark the **INSIDE** and **OUTSIDE** face of each piece so that I don't have to think about that later. The inside/outside orientation matters for every cut. I make all of these markings right along each edge where the lock miter pattern goes, just beyond the cutting area (about 3/4" from the edge), to help make sure I'm cutting the correct edge. I don't want to have to think too much while the router is running - less chance of screwing something up.

Install a 1/4" straight bit in your router. This bit will be used for all routing throughout the procedure.

Set up an auxiliary fence for your table saw. Simply cut a piece of 3/4" MDF or plywood to roughly the same size as your saw's fence, and attach it to the blade side of the fence with clamps, with the bottom of the aux fence about 3/8" above the table top.

A few tips, tricks, and precautions

Always unplug the router before adjusting the bit height or fence position.

Make multiple router passes for grooves deeper than 1/4". Router bits work best doing a little at a time. For deep grooves, start with a first pass at 1/4" deep, then repeat at 1/2" deep, and so on until reaching the target depth. Don't move the fence between passes - only change the bit height.

Take a close look at your router bit and notice how it's not perfectly round - it probably has a slightly oval shape, with cutting edges that stick out slightly. When you're aligning something with the outside of the bit, you want to make sure you're using the widest part of the cutting edge as the reference point, since the bit will remove material out to the widest point.

Take a close look at your table saw blade and notice how the teeth are a bit wider than the main disk. When aligning an edge with the saw blade, you always want to align with the widest point of the teeth, since the cut line will go out that far.

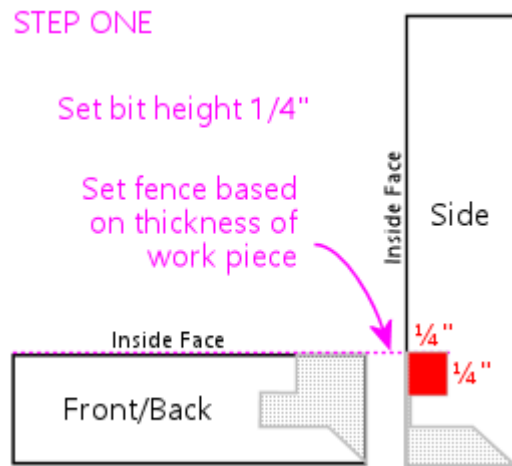
To avoid tear-out at the end of each cut where the router bit exits the board, try putting some masking tape along the trailing edge. If that doesn't work, use a scrap piece of the same 3/4" material, and keep it pressed up against the trailing edge as you push the end of the real work piece past the bit.

Cutting steps

The cutting procedure works by aligning each cut with a previous cut. I've tried to provide an easy method to hit each reference point without any guesswork. At each step, I'll explain the goal we're trying to accomplish, and then give you the method I use to accomplish it. Each step also starts with a schematic diagram showing the goal, but don't worry about trying to puzzle out the setup from the schematic - it's just there so you can see what the step's cut will look like and how it fits into the overall pattern.

Several of the steps have a setup procedure *and* a test procedure. The test procedure in each step is optional, so feel free to skip that part if it feels too tedious. The setup steps alone are pretty reliable. But I find that the extra test-and-adjust steps help me get the alignments closer to perfect.

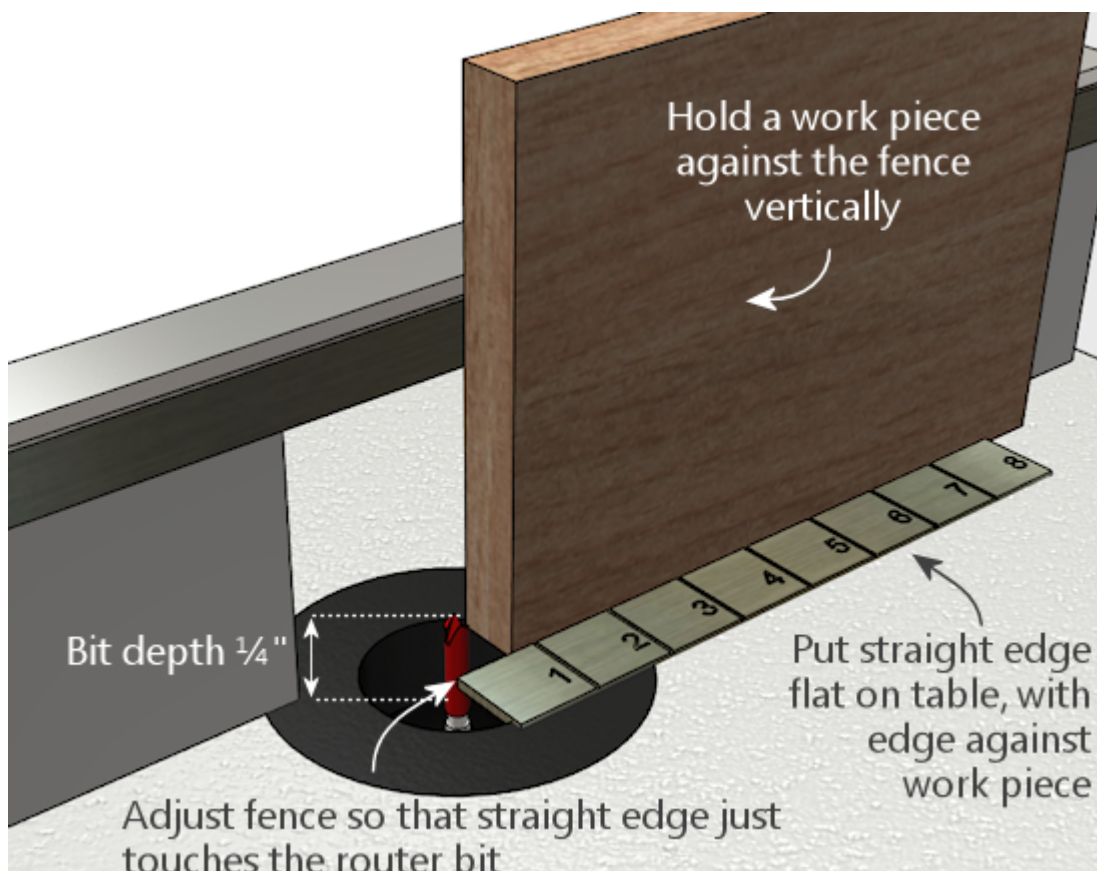
Step 1: Inner groove on the side pieces. This groove has to align with the inside face of the front/back piece, so we'll use the thickness of a sample work piece as our alignment guide for the fence. The bit height is simply 1/4".

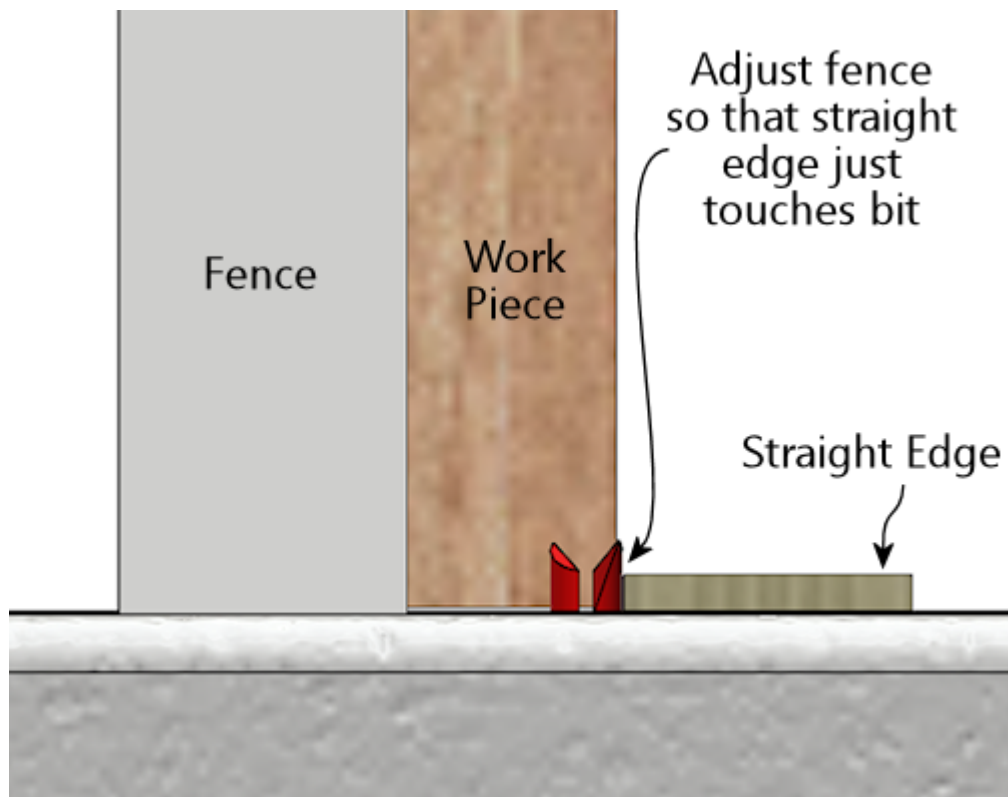


Set the bit height to 1/4" above the table, using a ruler or other measuring tool of your choice. (For better precision, use good set of calibrated setup bars - search for **router setup bars** on Amazon.)

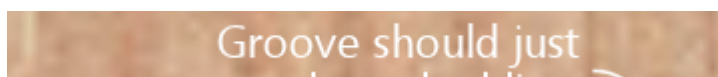
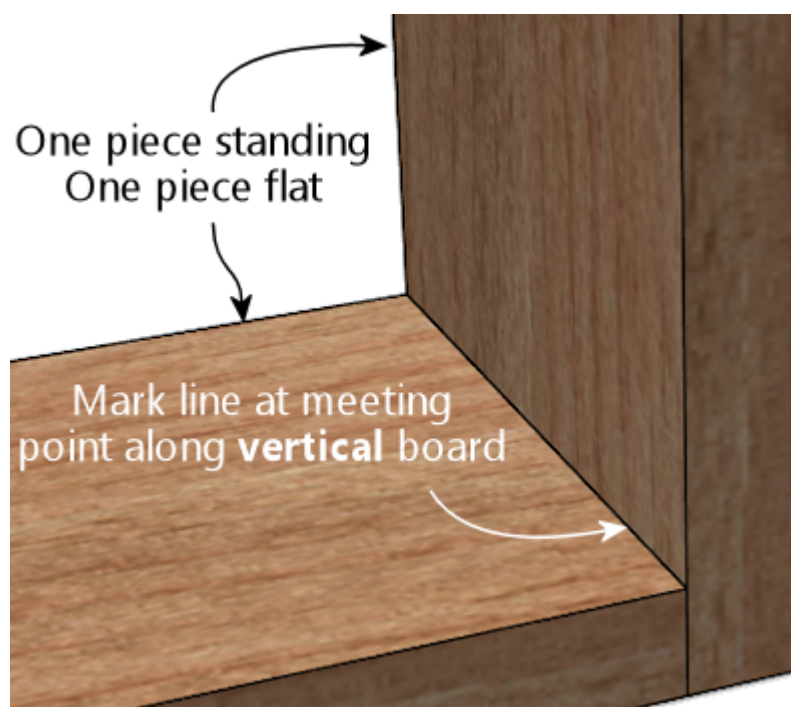
To position the fence, place one of the work pieces against the fence **vertically**. (It doesn't matter which piece - all we care about here is the thickness, which is the same for all work pieces.) Make sure to keep it tight against the fence. Place a straight edge flat on the table along the edge of the work piece, with one end extending out so that it's alongside the bit. Adjust the fence so that the bit just barely touches the straight edge. Rotate the bit back and forth to make sure that you're testing it at the widest point.

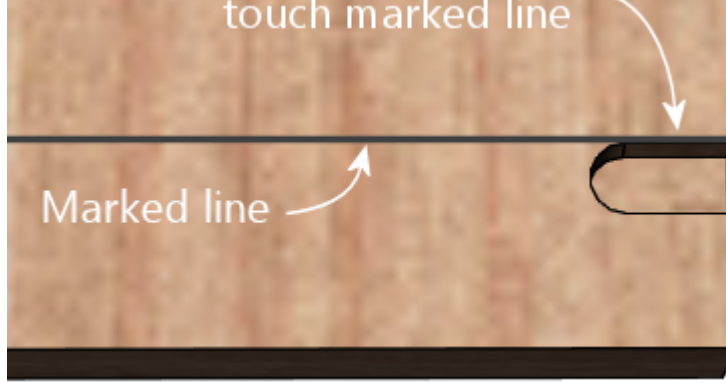
This is a good time to deploy the fence micro-adjuster, to help get the alignment perfect.



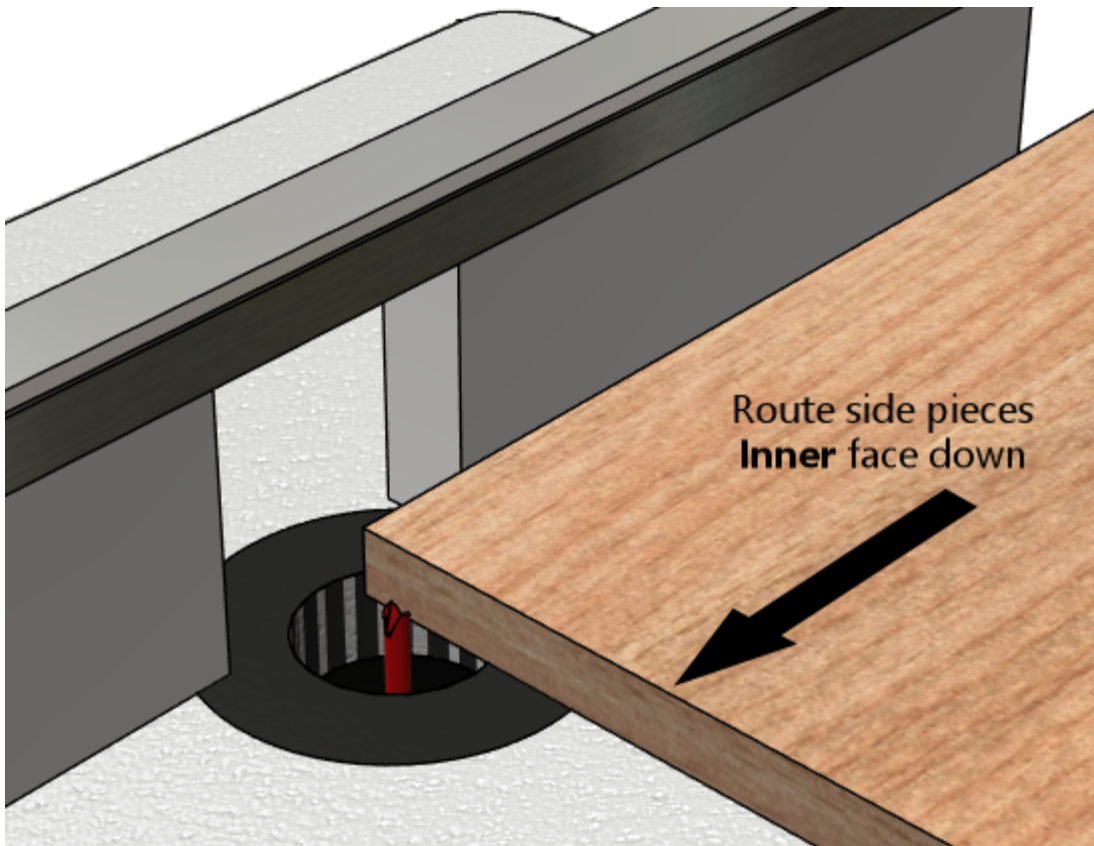


The alignment on this step is pretty important, so if you want to be extra-careful, you can run a quick test. Take two scrap pieces (from the same plywood as the real pieces, of course), lay one flat, and stand the other one up alongside it as shown below. Mark a line on the vertical piece using the horizontal piece as a straight edge. Route just an inch or so of the marked piece (line facing down and towards the fence), and check that the groove is exactly up against the line. It shouldn't cross the line or cut into the line - it should leave the line intact and just touch it with exactly zero clearance. If it's a little off one way or the other, use the micro-adjuster to compensate, and test it again.





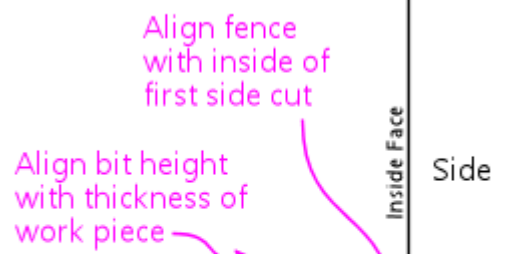
Once you're set up, it's time to do the actual routing for this step. Route each **SIDE** piece, with the **INSIDE FACE** down.

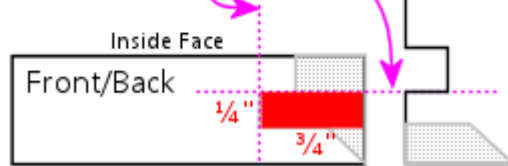


Remember to route **both ends** of each side piece, for the joins at the front **and** back.

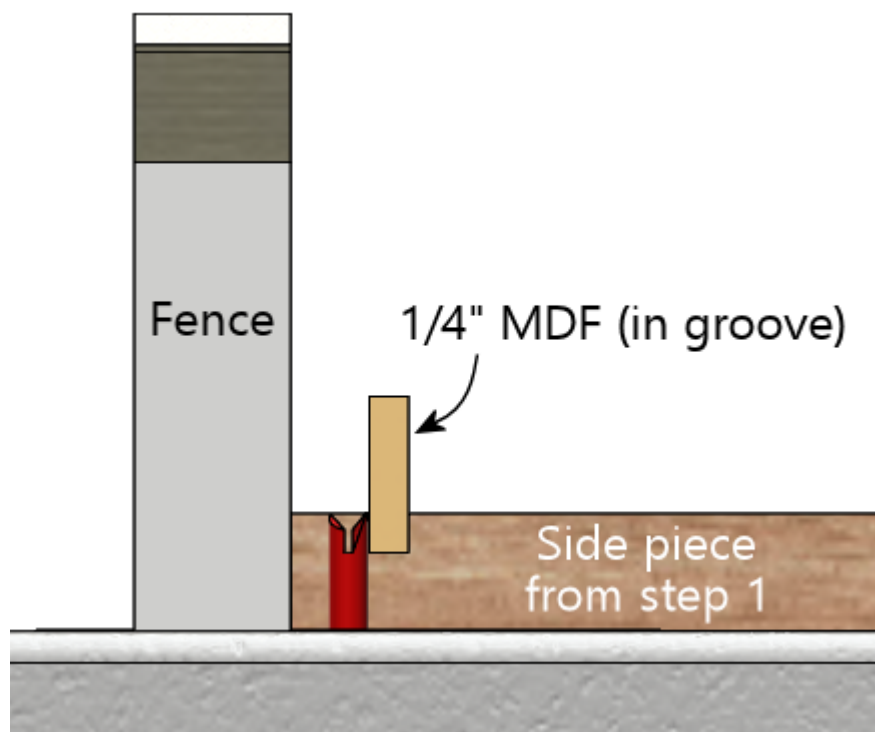
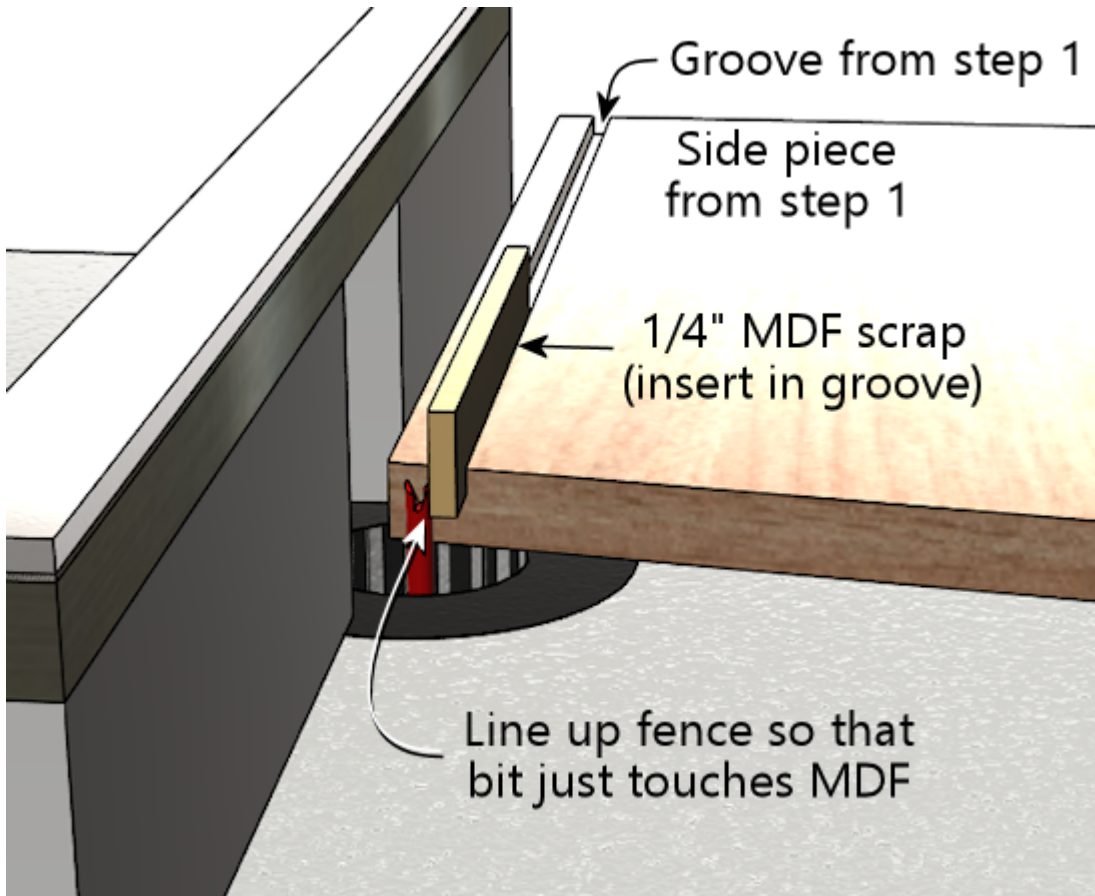
Step 2: Cut the center groove in the front/back pieces. The groove's depth has to equal the thickness of the work piece, so we'll set the bit height based on a sample work piece's thickness. The inside of the groove has to align with the outside of the side groove from step 1, so we'll set the fence position using the actual groove in one of the side pieces as a reference point.

STEP TWO

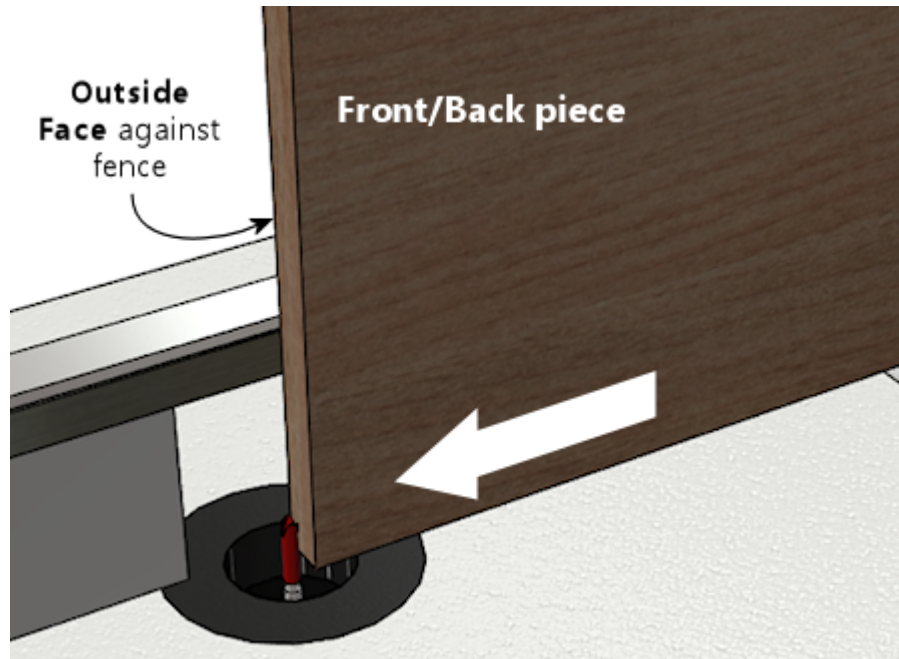




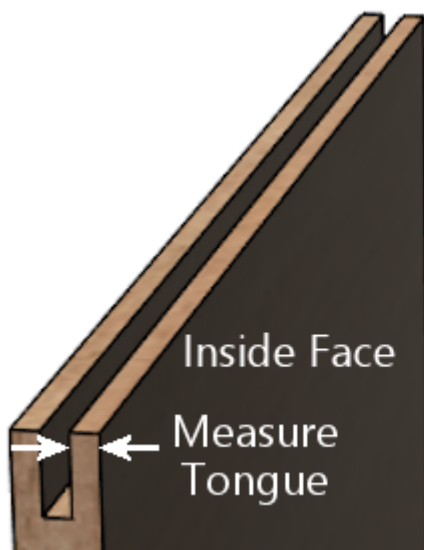
Grab one of the side pieces you just cut and fit a piece of 1/4" MDF scrap into the groove as shown below. Place it on the router table with the edge against the fence. Set the fence so that the bit just touches the MDF.



Here's how we're going to do the routing for this step. Route the **FRONT** and **BACK** pieces **vertically against the router fence**, with the **OUTSIDE FACE** to the fence. But we're not quite ready yet - we might want to check the alignment first, and we still need to set the bit height.



If you want to check the alignment to make sure it's right, there's a quick test that you can do before proceeding. Set the bit height to about 1/4" above the table, grab a scrap testing piece, and route it as shown above. This is just a test cut, so you only need to go an inch or two. Now measure the width of the "tongue" on the **inside** face, as illustrated below. I find digital calipers are the easiest way to get a precise reading here. The thickness should be just a hair under 1/4" (6.35mm) thick. Something like 6.2mm to 6.3mm is perfect. This tongue slots into the 1/4" groove from step 1 when you assemble the corner, so it can't be any thicker than 1/4", or it won't fit the groove. It's best if it's just the slightest bit thinner than 1/4", so that it fits easily but snugly. A looser fit isn't a disaster, but it won't be as strong when assembled, and it might not self-align as perfectly. If you're using the fence micro-adjustor, you can use the measurement to figure out how many turns of the screw are needed to get it exactly to a target of, say, 6.2mm. In any case, if you make an adjustment, I'd repeat the test (in my case, because I'm 50% likely to go the wrong direction, even if I'm looking straight at these instructions).



Goal: just barely under 1/4" (6.35mm)

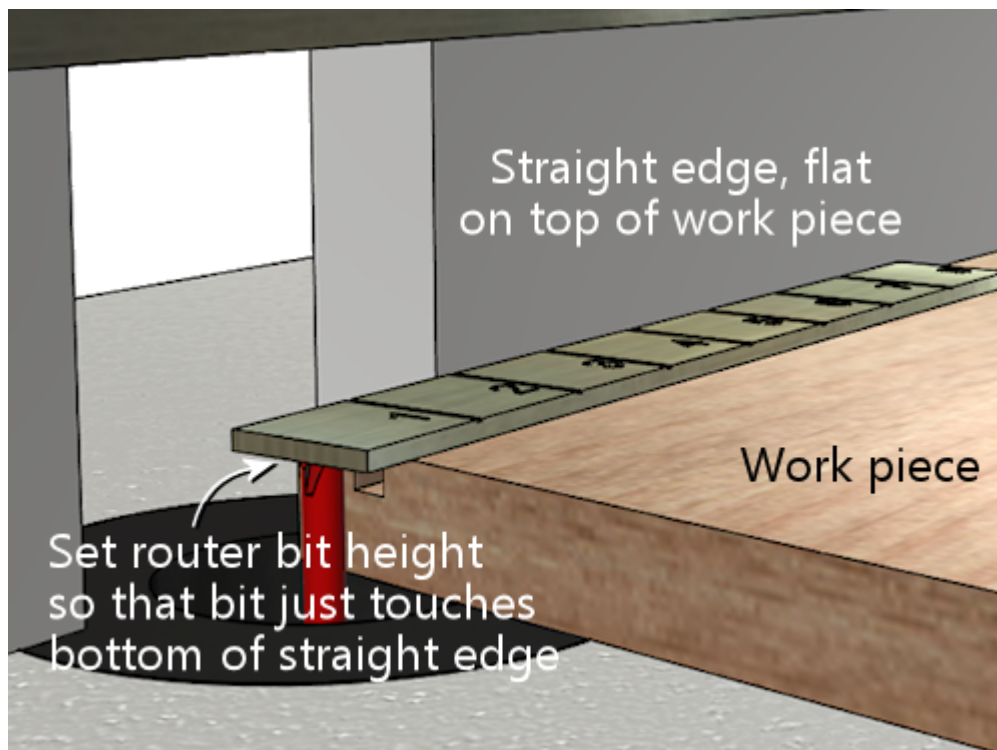
If over: move fence **back**
If under, move fence **forward**

Measured Width	Make This Adjustment
Over 1/4"	Move fence back (away from bit)
Under 1/4"	Move fence forward (towards bit)

Now we're ready to set the bit height. This is a deep enough groove that it's best to route it in multiple passes, so that we don't try to force the router bit through too much wood at one time.

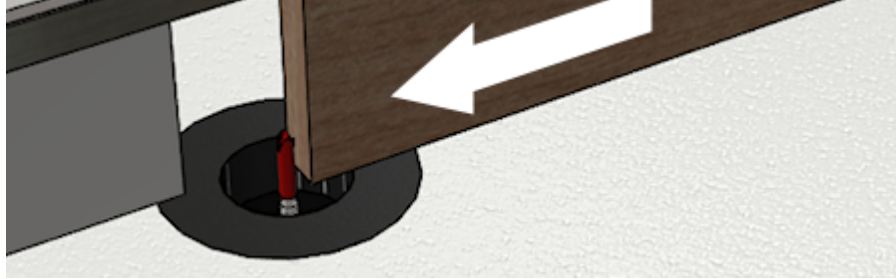
- First pass: bit height set to roughly 1/4" above the table
- Second pass: bit height set to roughly 1/2" above the table
- Final pass: calibrate to the work piece thickness

For the final pass, calibrate the bit height to match the thickness of the work pieces by placing a work piece flat on the table alongside the router bit. Adjust the bit height so that it matches the plywood thickness (or just a hair beyond). A straight edge on top of the work piece can help get it perfectly aligned.

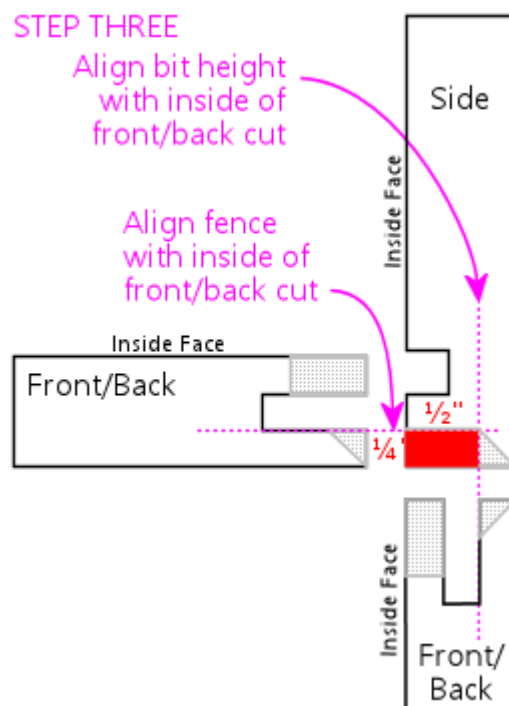


For each pass, route as shown in the illustration we saw earlier: route the **FRONT** and **BACK** pieces **vertically against the router fence**, with the **OUTSIDE FACE** to the fence. Here's the illustration again:

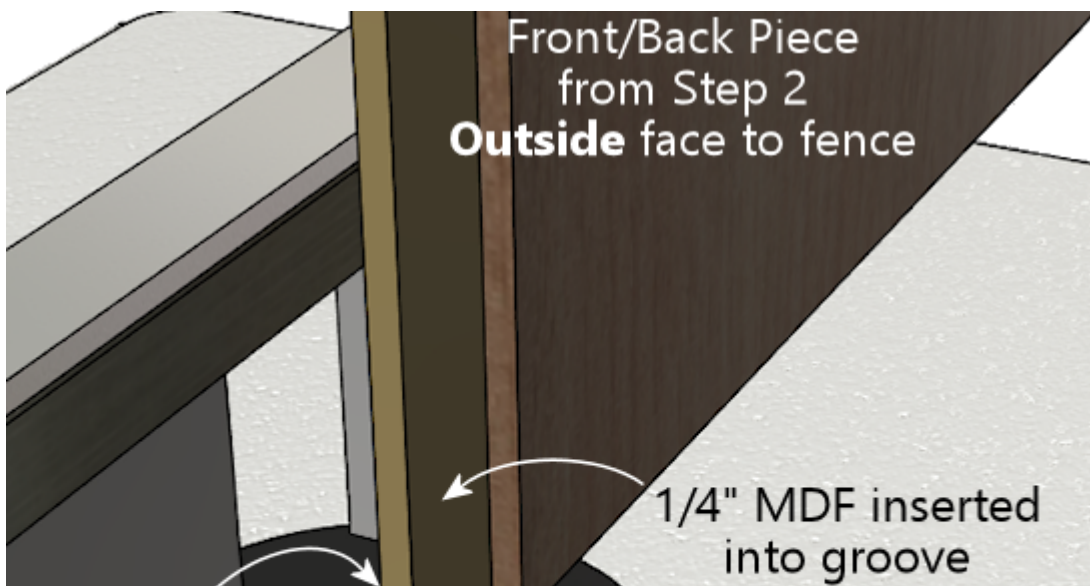


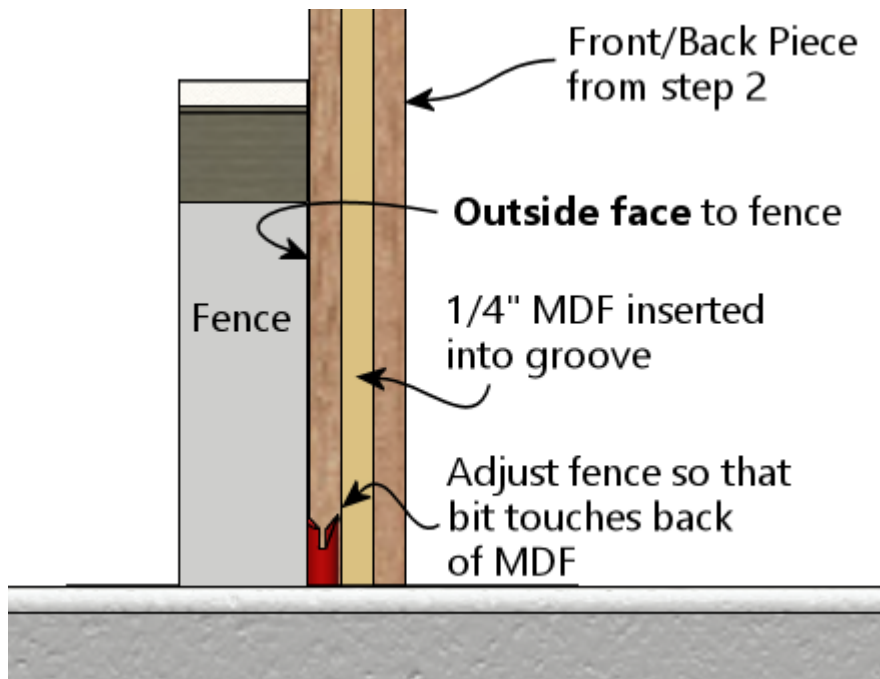


Step 3: Outer groove on the side pieces. This step finishes the "tongue" on the side pieces. The fence position for this cut has to align with the outer side of the front/back groove from step 2, so we'll use a front/back piece from the previous step as a reference point. The bit height has to align to the leftover portion outside the groove from step 2, so we'll also use a front/back piece as a reference for that.



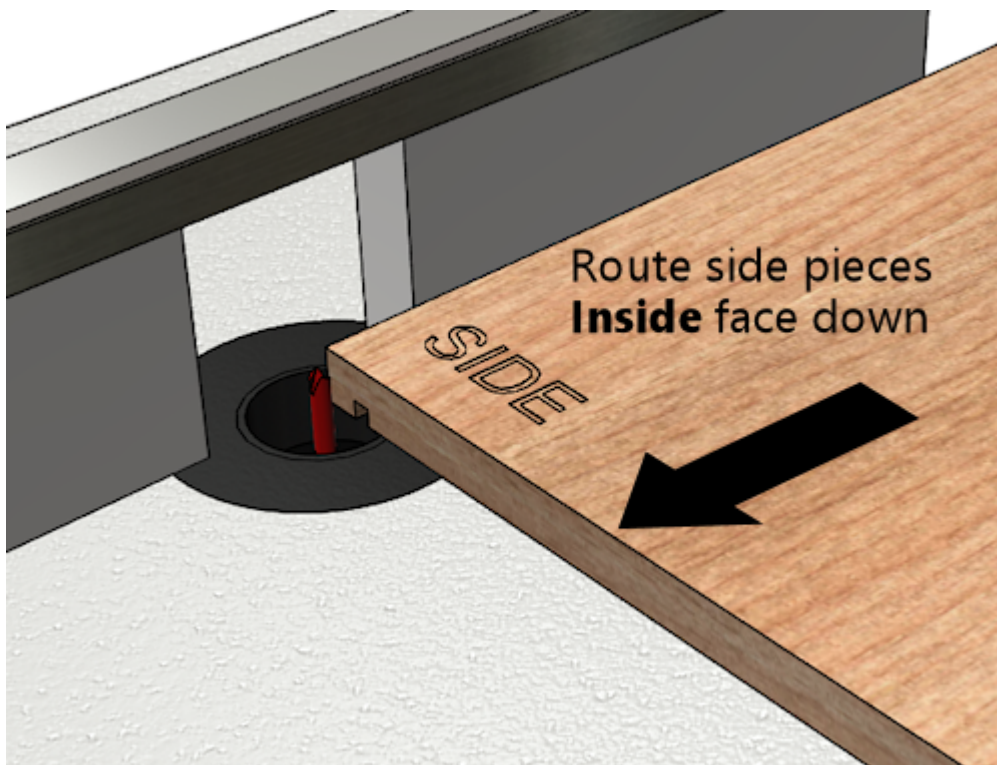
Take a front/back piece from step 2, and insert a scrap piece of 1/4" MDF into the groove. Stand this vertically against the router fence with the **OUTSIDE** face of the front/back piece against the fence. Place it so that the MDF extends just past the bit. Adjust the fence so that the router bit just touches the back of the MDF.



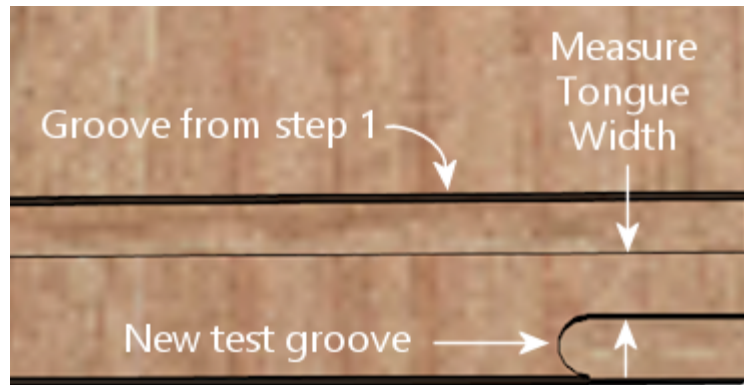


This is another deep cut that's best done in two passes - first at 1/4" deep, then at full depth. So set the bit height to 1/4" above the table for the first pass, route all of your side pieces, and then set the final bit height as described below.

Here's how we're going to do the routing: **SIDE** pieces, flat against the router table, **INSIDE** face down.



Before routing your actual side pieces, though, you can do a quick test to check the fence alignment. You'll need the scrap "side" piece from step 1, that already has the 1/4" wide by 1/4" deep groove set in 1/2" from the edge. Do the test on that piece. Route it as shown above, but you only need to go for an inch or two for this test. Now measure the width of the "tongue" between the two grooves:

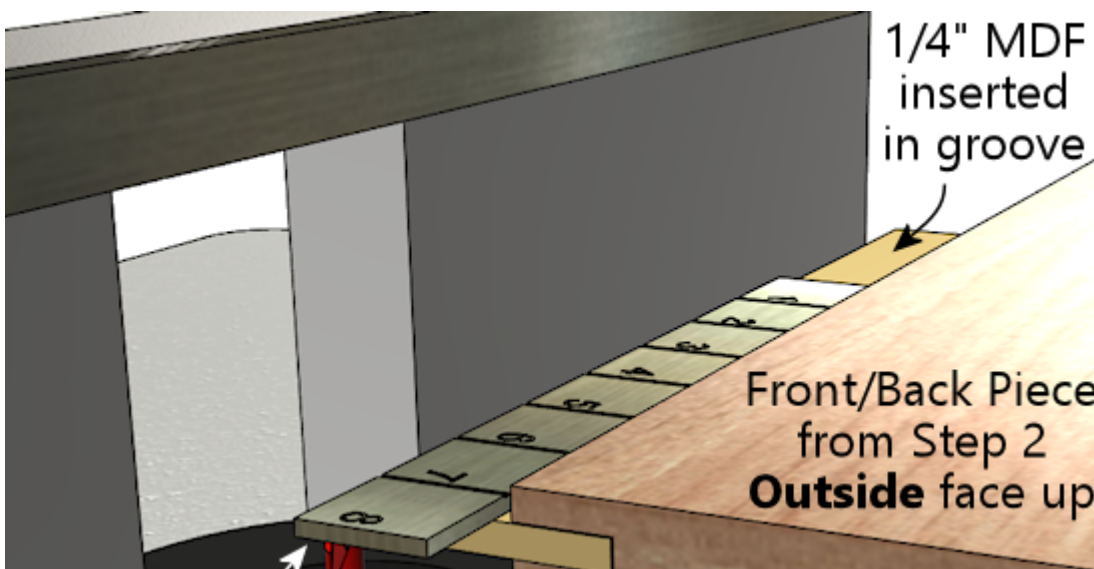


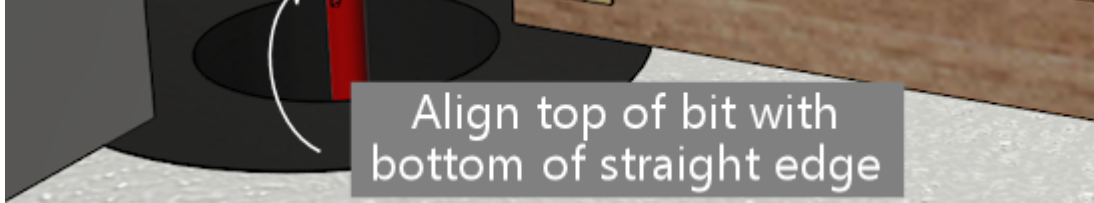
It should measure just under 1/4" (6.35mm). This part has to fit into a 1/4" groove, so if it's more than 1/4" wide, it won't fit. You don't it to be too loose, either, so the ideal is something just under 1/4", perhaps 6.2mm to 6.3mm. If it's too wide, move the fence back (away from the bit); if it's too narrow, move the fence forward (towards the bit).

Measured Width	Make This Adjustment
Over 1/4"	Move fence back (away from bit)
Under 1/4"	Move fence forward (towards bit)

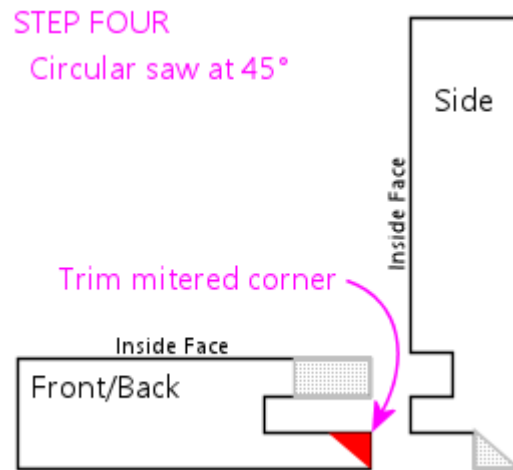
Once you're satisfied with the fence alignment, route all of the side pieces as shown above, keeping the bit height set at 1/4" above the table. Once done, you'll need to make one more pass over all of the pieces at the final bit height.

To set the final bit height, use a front/back piece with a piece of 1/4" MDF inserted into the groove. Put the whole thing flat onto the router table as shown below, with the **OUTSIDE FACE** facing up. Put a straight edge on top of the MDF so that it extends out over the bit. Adjust the bit height so that the top of the bit just touches the bottom of the straight edge.



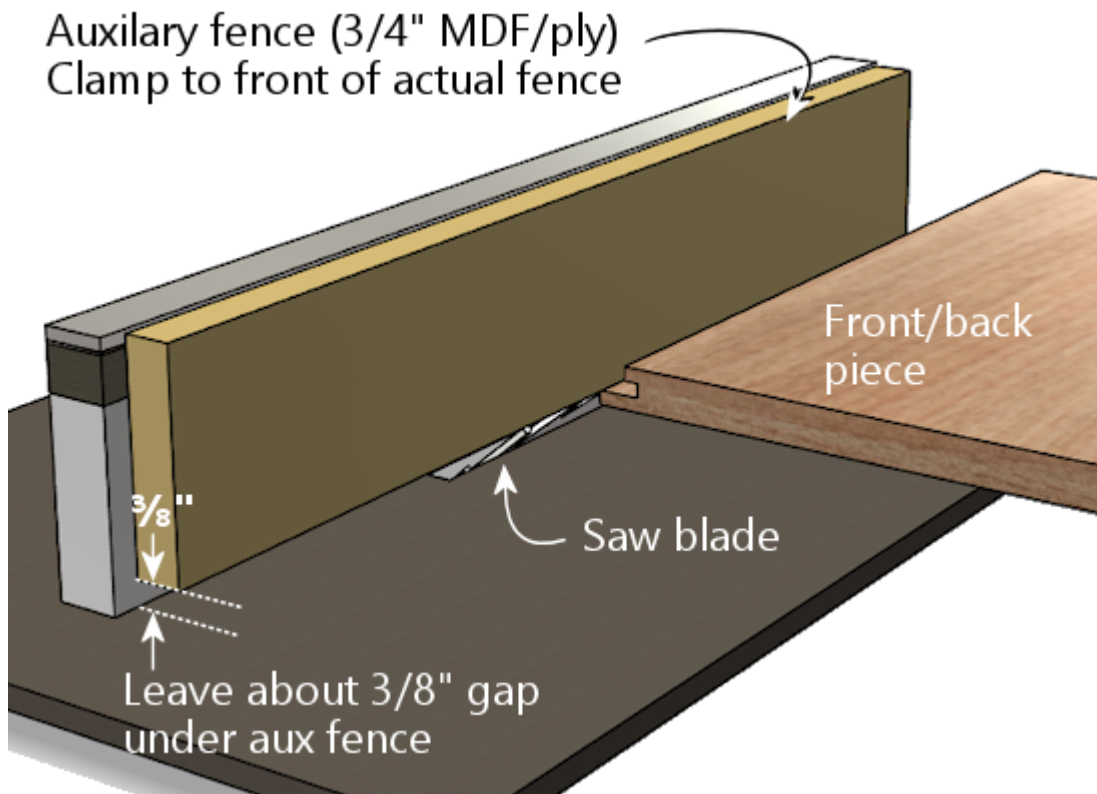


Step 4: Cut the 45° miter on the front pieces.

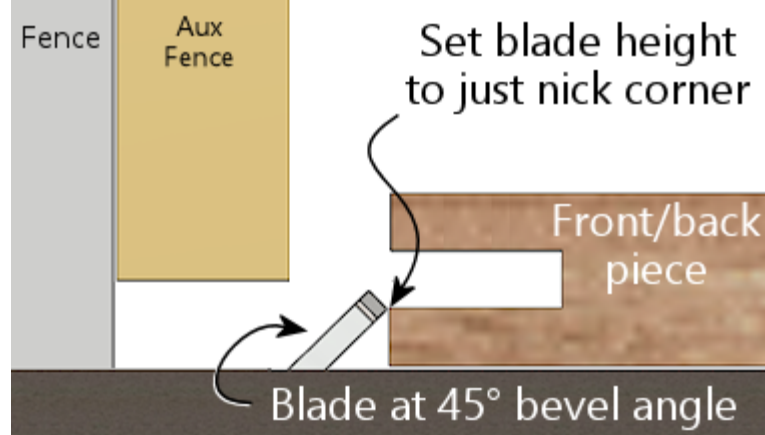


This step (and the rest of the steps) are on the table saw.

First, set up an auxiliary fence on your table saw. This is just a piece of 3/4" MDF or plywood, cut to about the size of the fence, and clamped to the front of the fence. Leave a gap of about 3/8" between the bottom of the auxiliary fence and the table top, to make room for the blade to extend 1/4" above the table.

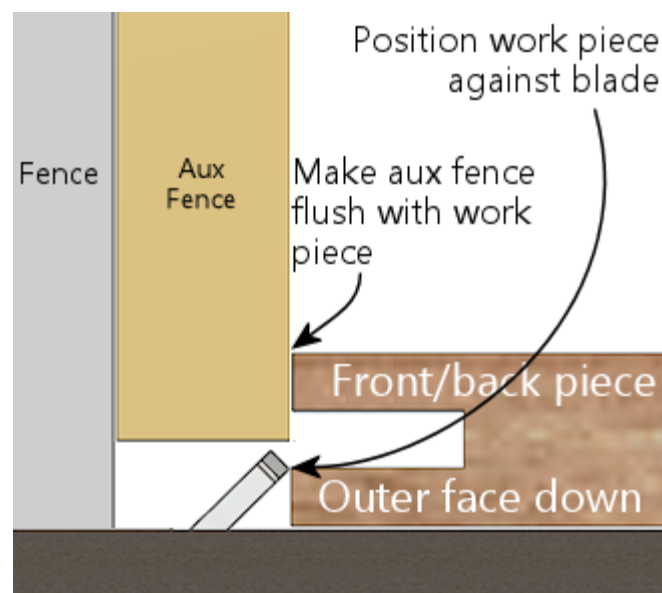


Set the saw blade to 45°. Adjust the height so that the blade just nicks the corner of the tongue in a front/back piece, as illustrated below.

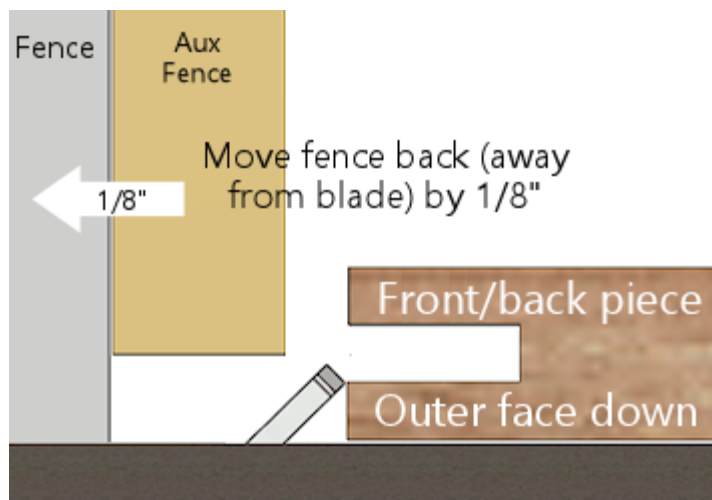


This cut needs to be precise, and I haven't found an easy trick like in previous steps to align it with another cut. The best approach I can come up with is to "sneak up" on the final cutting position with a series of test cuts. The good news is that you only have to do a series of test cuts for one piece. Once that one piece looks perfect, just run the rest of the front/back pieces through the saw with the identical fence position, and they should all come out equally perfect.

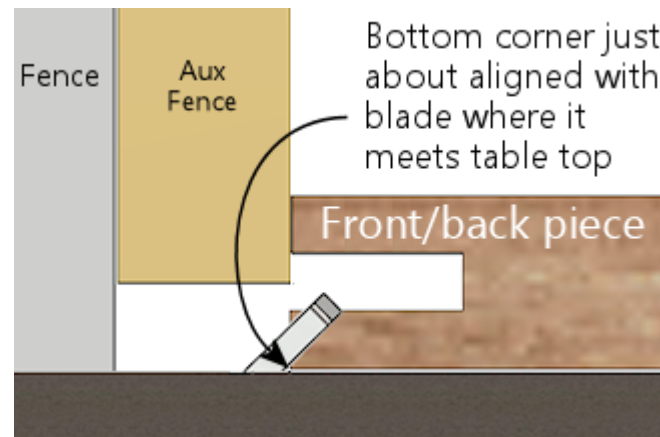
Start by positioning the fence to remove no material at all, by pushing the work piece right up against the blade, and making the aux fence flush with the work piece:



Now move fence back (away from the blade) by about 1/8".

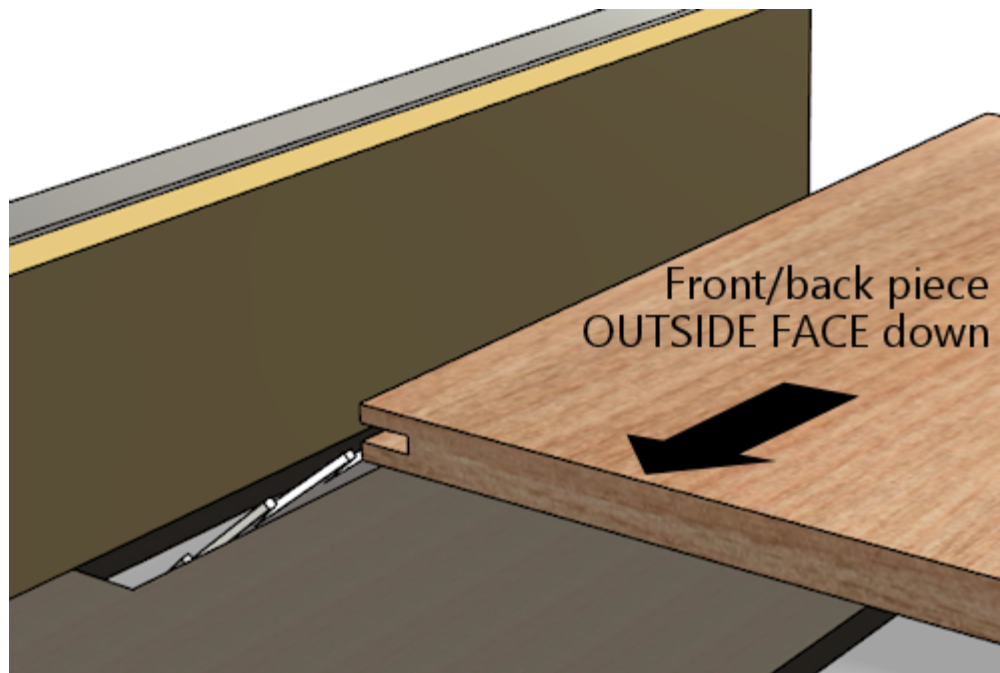


When you move the work piece back behind the blade and push it up against the fence, the bottom corner of the work piece should now just about line up with the saw blade:



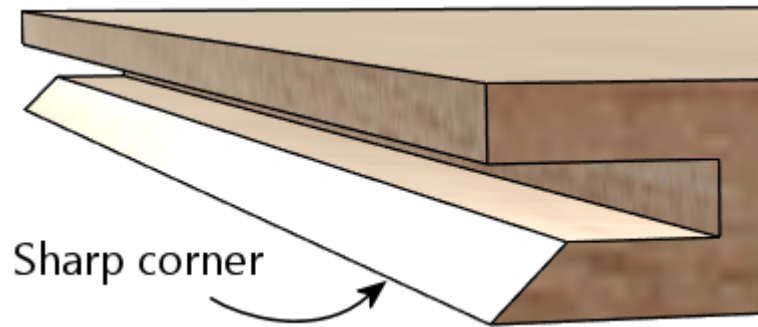
Double-check by eye that the blade is aligned as shown. Remember, it's best to remove **too little** material at this point than too much, since we can adjust things and take off a little more if we have to. That's why we only moved the fence 1/8", even though the ultimate target is closer to 1/4". If it looks like the blade is already going to hit the bottom corner, move the fence a little further towards the blade to open up a small gap.

Do a test cut, with the **front/back** piece **OUTSIDE FACE down**.



Tip: after each fence adjustment (including the first one), initially only run the first few inches of the board through the saw, and inspect the result to make sure the fence isn't already too back. If the fence is too far back, you'll be cutting material off the outside face of the outside edge, which you don't want to do. When this happens, you'll see a little indent along the outside edge where material was cut off. The goal is to get the 45° cut *exactly up to the outside edge of the outer face* without actually cutting any material off the outer face. Making a partial cut lets you see immediately if you've gone too far, since the outside edge will no longer be a straight line. You can't undo this, which is why I like to include a scrap piece in every batch for these first cuts while adjusting the fence.

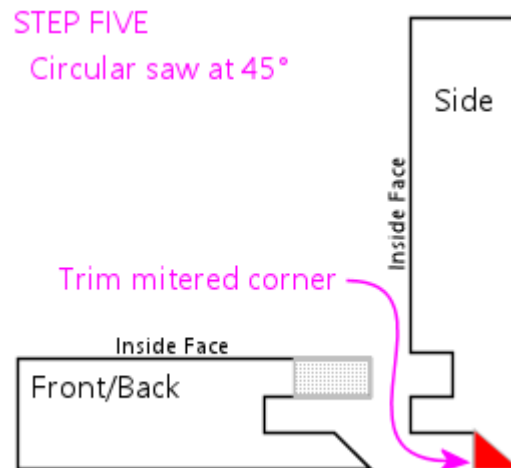
When it's exactly right, you should have a sharp outside corner.



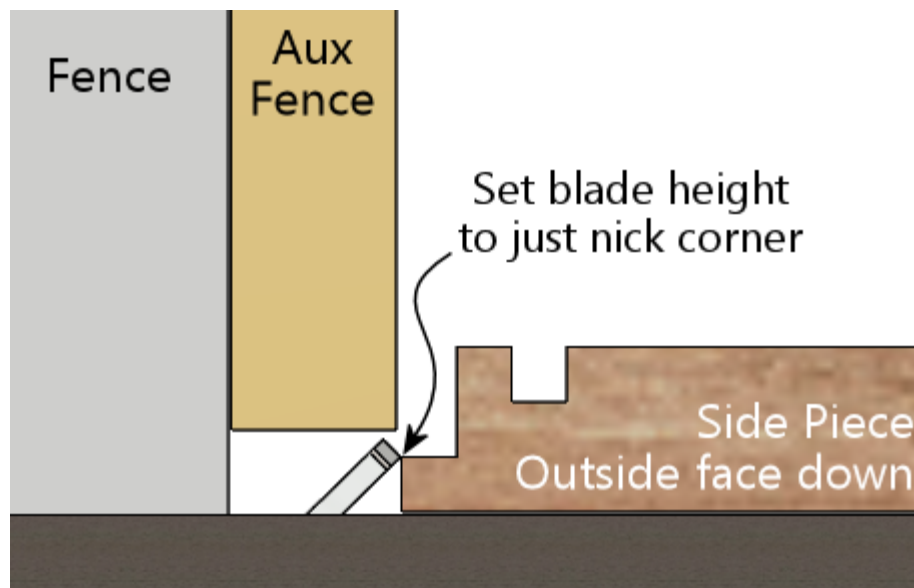
If the corner is still a little squared, you just need to adjust the fence back slightly (away from the blade) and try again. **Make small changes** to ensure you don't overshoot. Repeat until the corner is sharp.

You can now run all of the remaining front/back pieces through.

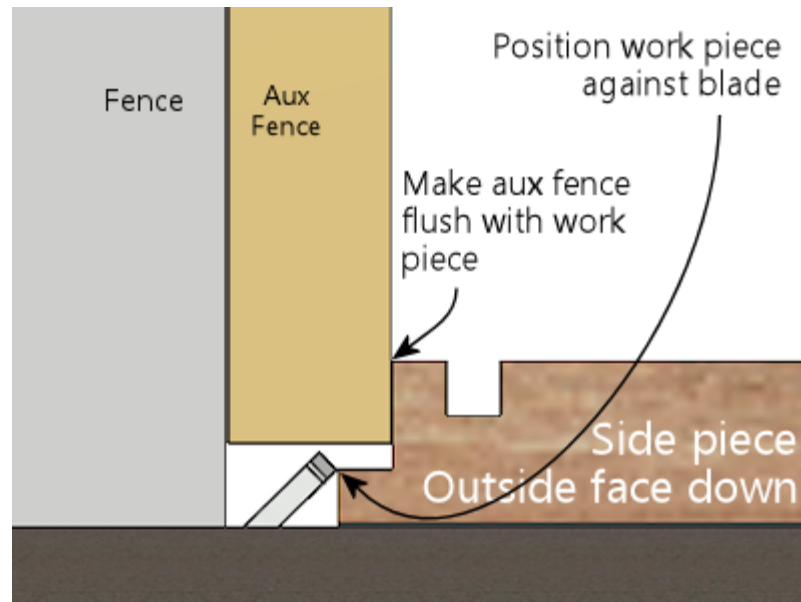
Step 5: Cut the 45° miter on the side pieces. This is essentially the same as step 5, just with a different blade position to match the side pieces this time.



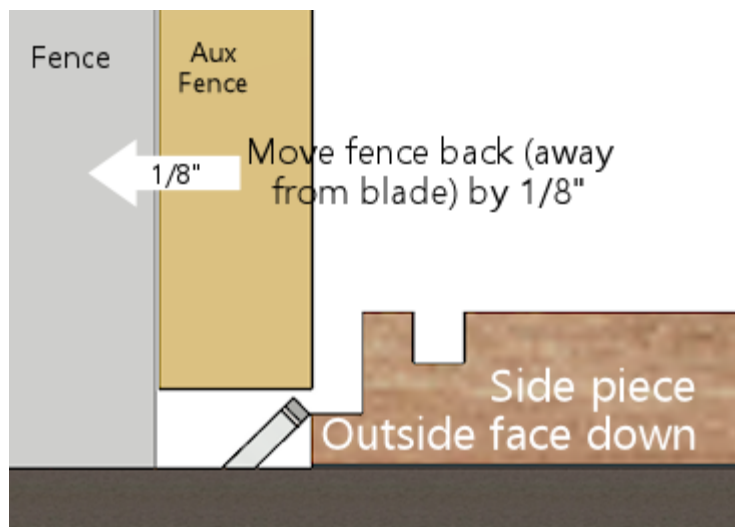
You should be able to keep the same blade height from the previous step, but check it with a side piece to confirm.



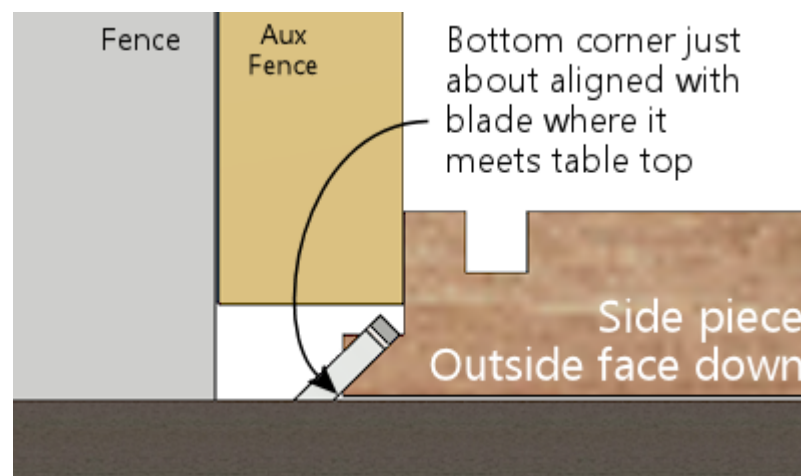
Adjust the fence position using the same drill as last time. Start by pushing the work piece up against the blade, and making the fence flush with the work piece:



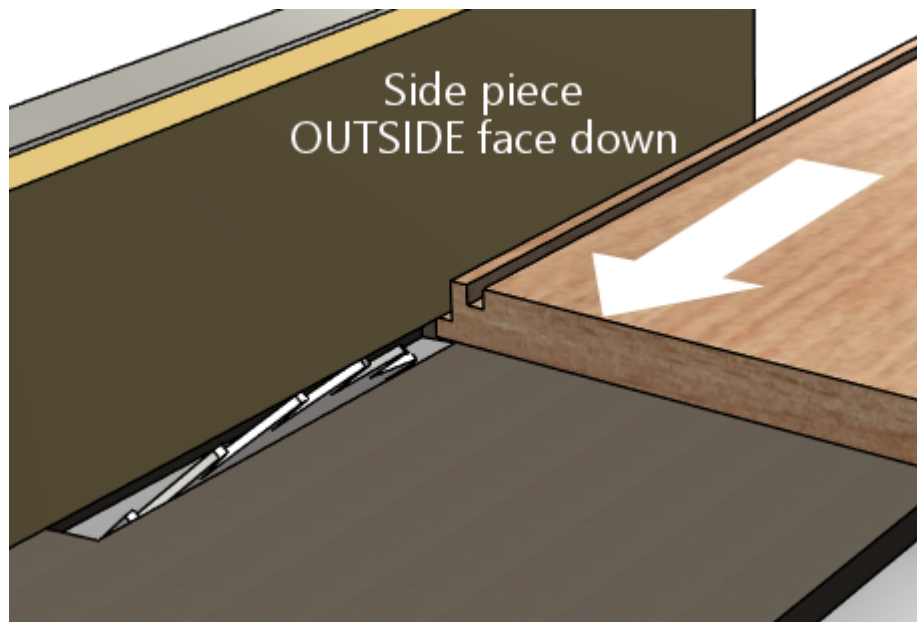
Now back the fence away from the blade by about 1/8".



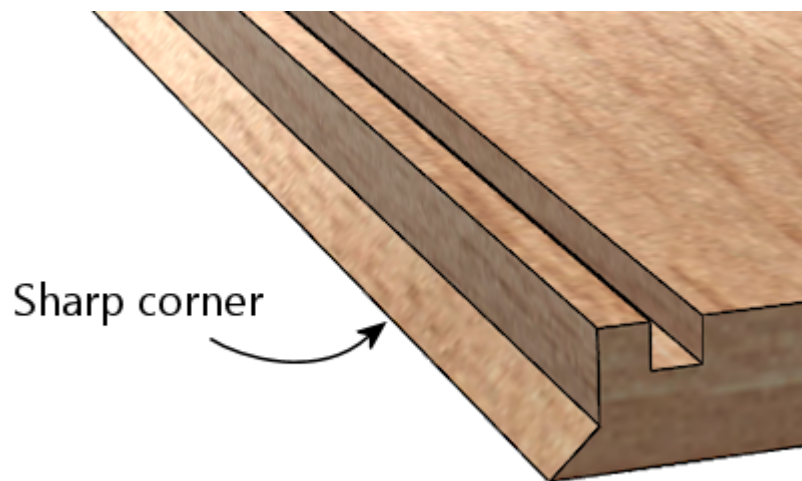
Move the work piece behind the blade and flush with the fence, and verify that the blade isn't quite going to cut hit the bottom corner. As before, it's better to cut too little at this point, since we can re-cut a little more if necessary.



Do a test cut, with the **side** piece **OUTSIDE FACE down**.



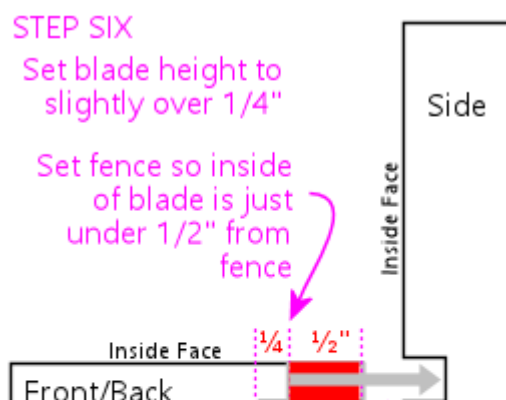
As before, we're looking for a nice sharp corner.



If the corner is still a little squared, adjust the fence back slightly (away from the blade) and try again. **Make small changes** to avoid overshooting. Repeat until the corner is sharp.

You can now run all of the remaining side pieces through.

Step 6: Trim the tongue on the inside of the front/back pieces so that it fits into the side slot from step 1.

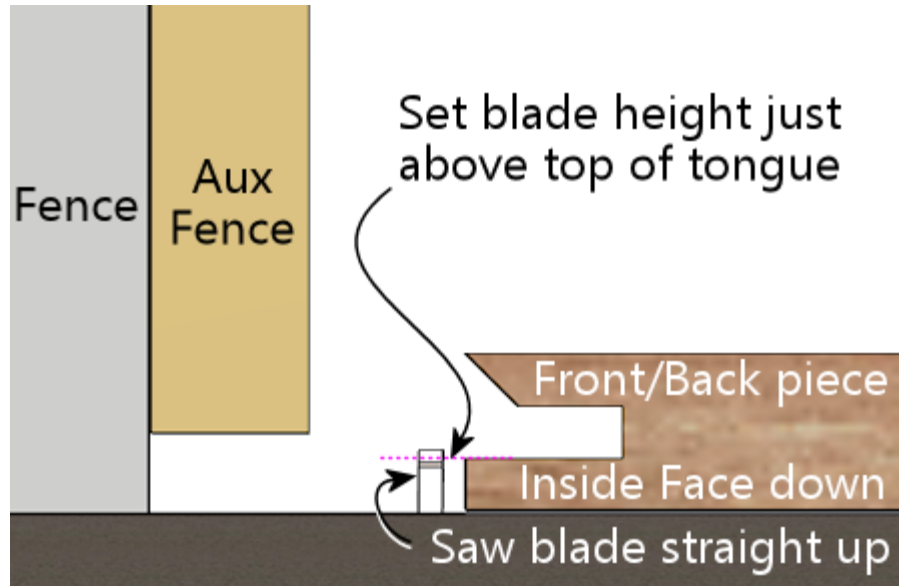




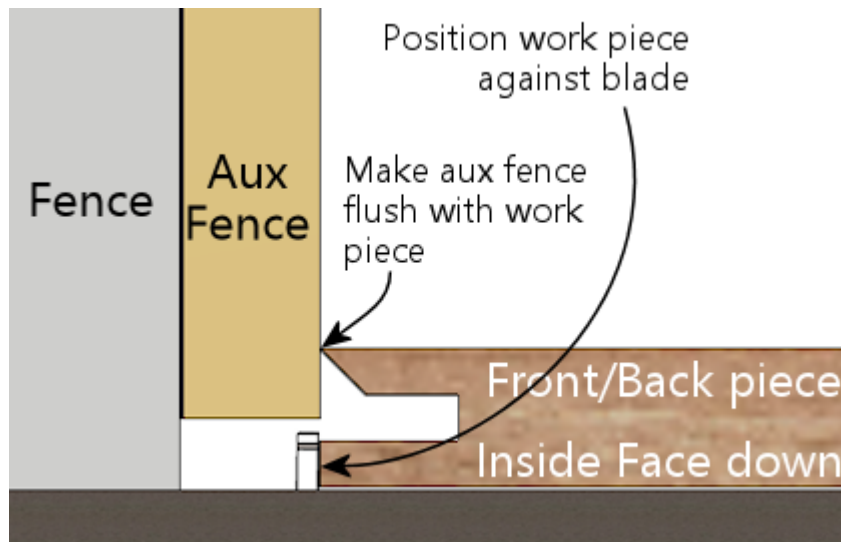
After cutting, test fit
Adjust fence & trim more
if needed for good fit

I do this step last because it's another "sneak up" cut, and doing it last lets us test to see if we trimmed enough off by doing an actual test fit. That's the best test for exact alignment.

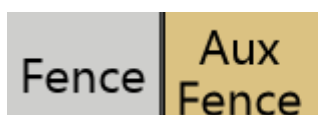
We'll use the table saw for this cut. Set the blade so that it's straight up. Position a front/back piece on the table next to the blade, inside face down. Adjust the blade height so that it's just above the top of the tongue - about 1/4".

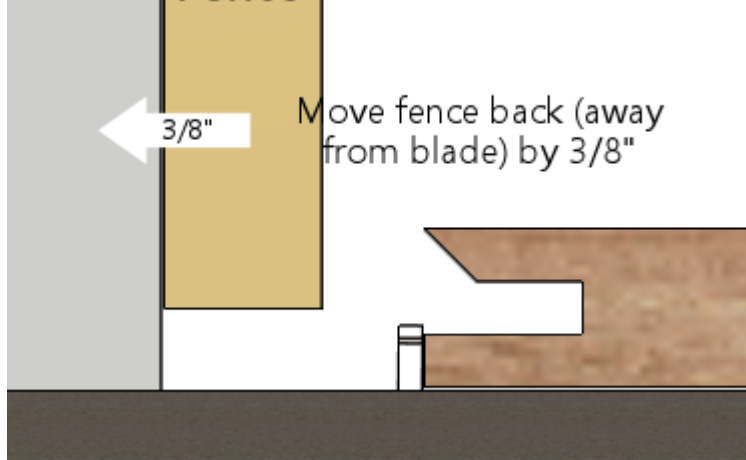


Put the work piece flush against the saw blade, and move the fence so that it's flush with the work piece.



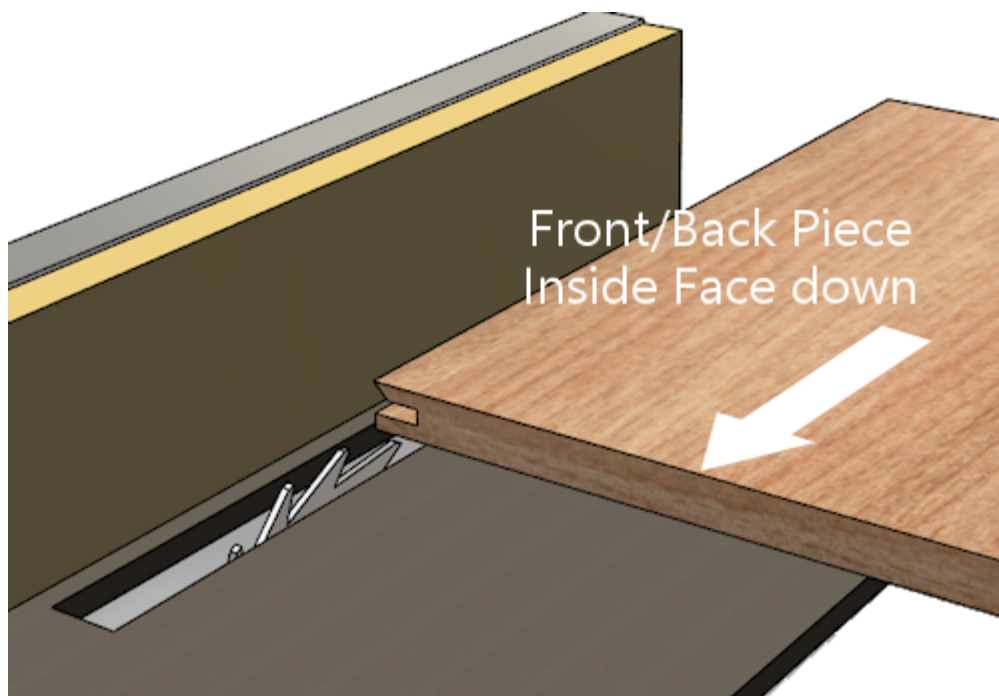
Now back off the fence, away from the blade, by about 3/8".





Since we have to "sneak up" on the final fence position with test cuts, just do one edge of one front/back piece at first. We'll test the fit and adjust as needed before running the other pieces.

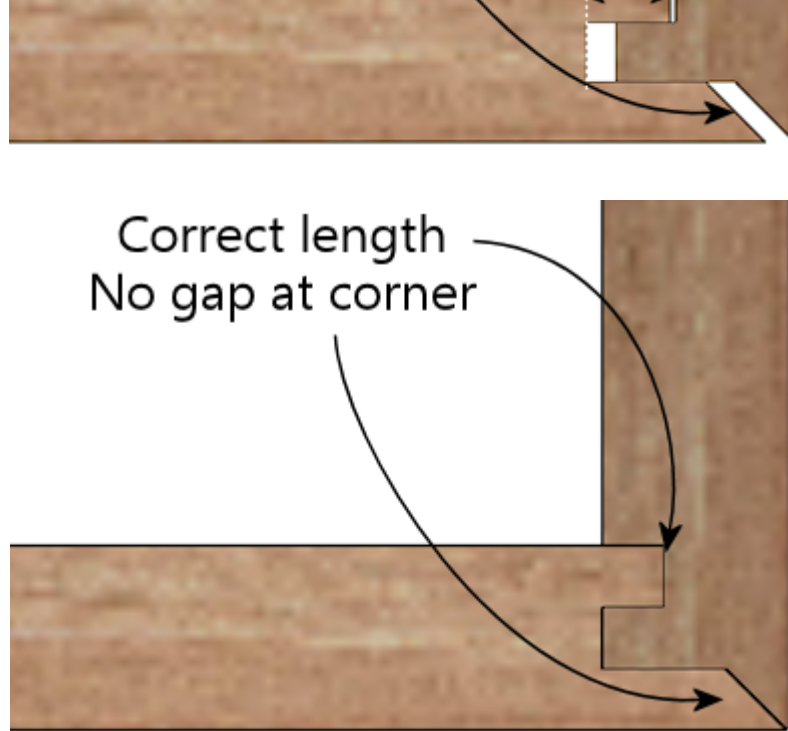
Run a **FRONT/BACK** piece through the saw, **INSIDE FACE down**.



Test the fit against a side piece. The two pieces should fit together at the outside corner without any gaps. If the tongue in the front/back piece is jutting out and preventing the pieces from fitting together, we need to trim it back a little more. Back the fence away from the blade - just slightly - and do another test cut. Repeat until it's a good fit.

Tongue is too long
Leaves a gap at corner

This diagram shows a close-up of the corner where two wooden pieces meet. A curved arrow points to the tongue of the front/back piece, which is jutting out and creating a gap at the corner. The text "Tongue is too long" and "Leaves a gap at corner" is written above the arrow.



Once you have a good fit with one piece, finish the remaining FRONT/BACK pieces.

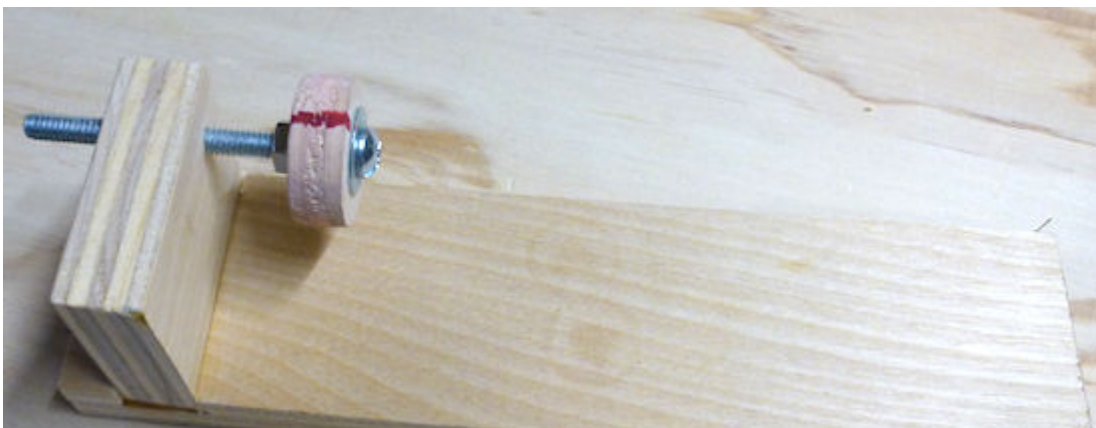
That's it - you're done! The corners should all fit together tightly. If any of them are so tight that you have to force them together, try manually sanding the sticking points.

Router fence micro-adjuster

Table saws usually come with rack-and-pinion fences that you can adjust fairly precisely by turning a knob. Router tables, in contrast, usually have much simpler fences, without any adjustment mechanism at all. The fence usually just slides freely on a couple of parallel tracks. You move it by hand to where you want it, then tighten some knobs that hold it in place.

That's not a very precise way of positioning the fence, but it's good enough for most applications. There are tasks where you can really benefit from more precise positioning, though. Lock mitering is one of them.

There's a solution, and it doesn't even require spending a fortune on a super-deluxe router table. You can make a really simple micro-adjuster for your existing router table fence. You don't have to make any permanent modifications to the router table, and you can build the micro-adjuster out of scrap wood and hardware you probably have lying around. Here's the one I built:





The wood is all 3/4" plywood, and the two pieces are simply glued together at a 90° angle. (I routed a shallow 3/4"-wide dado that the vertical piece slots into, which makes the glue joint nearly unbreakable.) The screw is a 3" long, 1/4"-20 machine screw. In the vertical plywood piece, I drilled a hole slightly smaller than 1/4", and then worked the screw in so that it cut its own threads into the wood.

I added a DIY knob, in the form of a "donut hole" cut out of 3/4" plywood using a 1" hole saw. (There's a KEPS lock nut on the inside to keep it in place.) I drew a big red mark so that I could easily gauge full turns, half turns, quarter turns, etc.

Alternatively, you can buy a plastic knob at Home Depot or Amazon, or skip the knob entirely and make adjustments with a screwdriver.

The micro-adjuster is simple to use. First, set up your fence as close as you can get "by hand" to where you want it. Second, tighten the knob-screw on one side to pin the fence down on that end. Third, position the micro-adjuster behind the fence on the *other* side (the un-pinned side), with the screw just touching the back of the fence. Clamp the adjuster to the router table so that it stays put at this position. You can now tighten the fence knob on this side to fix the fence position here.



The micro-adjuster as deployed, sitting behind the fence on my router table. It's simply clamped onto the table with an F-clamp. My fence is MDF, so I masking-taped a penny to the back where the screw contacts the fence, to make a harder contact point and prevent the screw from making a divot in the MDF. Note that I

also clamped a small wood block behind the fence on the opposite side to set a hard stop, to ensure that the fence can't slip back even slightly on that side. The fence already has a knob that locks it down on each side, but I wanted to ensure that there's absolutely no slippage when we micro-adjust the micro-adjuster side.

Now you can make tiny, precise adjustments to the fence position. When you need to move the fence by a tiny amount, loosen the fence knob on the micro-adjuster side only, and turn the screw in the appropriate direction to move the fence forward or back. When moving it back, you'll have to push the fence against the screw, since the screw won't pull it back on its own, but the adjustment should nonetheless stay very precise as long as you keep the fence pressed against the screw.

If you're using a 1/4"-20 screw like I am, each full 360° turn of the screw corresponds to 1/20 of an inch change in the fence position. (That's the "-20" in 1/4"-20 - it means that the screw has 20 threads per inch.) With a knob, it's easy to gauge a quarter turn, which corresponds to 1/80 of an inch, or even an eighth turn for 1/160". What's more, that's the distance the *screw* moves. The thing that matters is how far the center of the fence moves, since that's where the bit is. The center of the fence only moves half as far as the screw, since the screw is at one end and the other end is kept fixed in place. So an eighth turn on the knob equals 1/320" of travel at the center of the fence. That's plenty of precision for anything I've ever attempted.

You might wonder how we keep the fence "square" if we're only moving it at one end. Moving just one end obviously will cause the fence to angle slightly on each adjustment. If this were a table saw, that would be a huge problem. But it's okay with a router table! The difference is that router bits are round, so there's really nothing to be "square" to. (Okay, technically, table saw blades are also round, but a saw blade defines a plane that the fence must be parallel to. There's no equivalent parallel plane for a router bit, so it doesn't matter how the fence is angled.)

Appendix 13. Lock Miter II: The Special Router Bit Way



A lock miter bit set up in a router table, and a sample piece of 3/4" plywood showing the pattern that the bit cuts in the edge of the wood.

This section covers making a lock miter corner join using a special router bit purpose-built for this join. For an introduction to lock miter joins in general, see the previous section, Appendix 12, Lock Miter I: The Plywood-Friendly Way, which also describes a different way of making this type of join using just a straight router bit and a table saw.

I get the impression that practically everyone implementing lock miters these days uses the special bits, rather than the more manual approach of the previous section. It's easy to see why. The router bits only require one pass over each edge, compared with about six separate cuts per corner for the manual method. There's less risk of screwing up one of the steps, or cutting one of them slightly out of alignment, if you only have to make one cut in the first place. But there's a catch - actually, a few catches. One is that the bits are expensive (ranging from about \$25 for one that's cheap and priced accordingly, to over \$100 for a really nice one). Another catch is that they're notoriously difficult to set up properly. The reputation isn't undeserved; when you read through the setup procedure below, you might start to think that six cutting steps doesn't sound like so much trouble after all. Yet another issue is that lock miter bits don't always perform well with plywood (some people would rephrase that to "they always don't perform well with plywood"). The bits are really designed more for solid wood. They do work with plywood to some extent, but the small tongue-and-groove features don't always hold up well, and the veneer can get chipped up if you don't take some extra precautions (which we'll outline below).

You might wonder what's so different about the manual approach that makes it work so much better with plywood. The difference is the size of the tongue-and-groove features. The features are about twice as large when you use the manual approach. The smaller features in the router bit approach are just too delicate for most plywood. Follow-up question: why can't they make the router bit's features as big as the manual method's features? That's because the router bit has to cut a mirror-image pattern. That's only way that you can use the same router bit to cut both

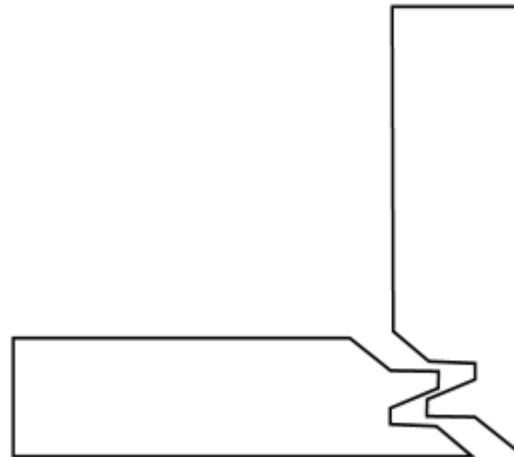
sides of each corner - one side has to be the mirror image of the other for the pieces to fit together. The mirror image means that every feature has to be there twice. And this all has to fit into the thickness of the board. That's why the features are half-sized compared to the manual method, where the cuts are complementary instead of mirrored.

I really like the idea of the lock miter bit, but I've found to my surprise that the "plywood-friendly" approach isn't any harder, and might even be a little bit easier. It's still a bit of work, but when you take into account the lengthy setup procedure for the lock miter bit, plus all of the extra precautions you have to take during the routing process, it might actually be more work overall with the special bit. Another practical drawback of the router bit method is that the bits are rather frighteningly large and probably more dangerous than typical router bits. Despite the disadvantages, though, the lock miter bit has the appeal that it should produce predictable and repeatable results after you get it calibrated properly. The plywood-friendly procedure involves more cuts, which I think is inherently scarier to a newbie woodworker like me, since there are more chances for something to go wrong.

If you do decide to try the lock miter bit approach, I'm hoping that the information in this section makes it at least a little easier for you to carry it out. Below you'll find a step-by-step recipe that I think is fairly complete and should produce good results if you follow it carefully. You can also find lots of Youtube videos about how to use lock miter bits, which I recommend seeking out - it's always good to see a new technique in action before trying it yourself, plus you'll get other perspectives on the right techniques. My procedure is almost the same as what everyone else recommends, but I think I've boiled it down to a more precise recipe than I've seen elsewhere. I've also made a couple of improvements to the usual setup advice that I think will help you get better accuracy with less work, and I've added tips about a couple of issues peculiar to pin cab construction.

How lock miter bits work

Lock miter bits achieve one-pass operation by using a symmetrical (mirror-image) pattern for the adjoining edges at each corner. You route one edge with the face horizontally on the router table, and you route the other edge with the face held vertically against the fence. The symmetry allows the two sides to mesh when joined at 90°.



The trick to making the corners align is to get the midpoint of the mirror-image pattern exactly at the center of each board. That's where all of the notorious setup difficulty comes from - you have to hit that center point almost perfectly for the

corners to align. And you have to do it in two dimensions simultaneously - bit height and fence depth.

The setup procedure is complex, but there's a pretty reliable recipe for it, and it doesn't require superhuman woodworking skills or feats of measurement. It's all based on iterative testing and adjusting, using a methodical approach that should (if all goes well) quickly converge on the right setup. Once you've dialed in the alignment, you just lock it down and route all of your boards with that same fixed setup. You don't have to make any further adjustments after the initial alignment - in fact, it's critical that you *don't* make any changes, since you want all of the cuts to be identical.

Does a lock miter bit even work with plywood?

A lot of woodworkers say no. In fact, the owner of VirtuaPin has mentioned that he doesn't use lock miters for the cabinets he sells because lock miters don't work well enough with the plywood stock he uses. He's undoubtedly talking about the lock miter bit rather than the plywood-friendly method, since that actually does work well with most good-quality plywood.

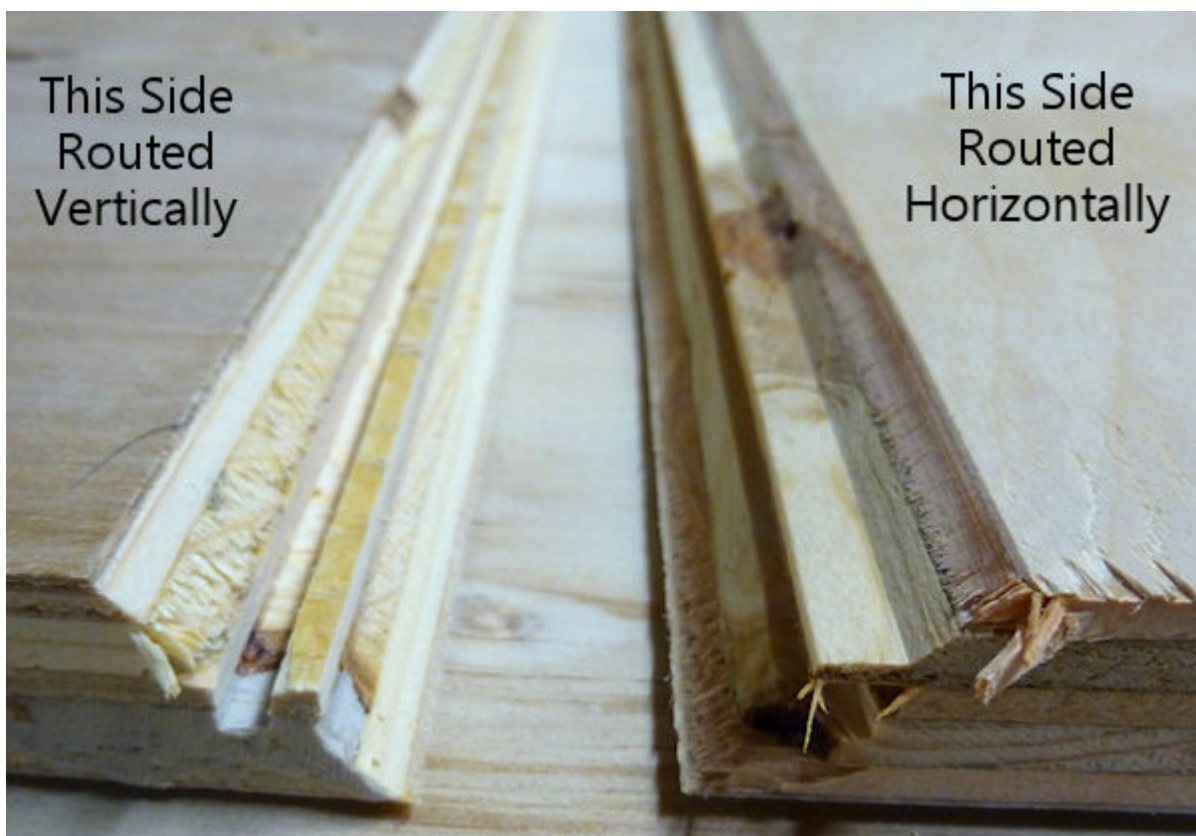
You might say, "Well, Williams used it, so it must work." It's true that Williams used a lock miter join on their 1990s cabinets, but they didn't do it with the mirror-image bits. If you look closely at their cabinets, you can see that they used the technique described in Appendix 12, Lock Miter I: The Plywood-Friendly Way instead.

My own experiences with the lock miter bit have been mostly successful. I've used it with mid-grade plywood from Home Depot for a few projects, and it works, with some reservations. To illustrate the kind of results I've seen, here are some photos from my "vertical sled" build (which has its own section later in this chapter).





Sample of lock miter results, using 3/4" birch plywood from Home Depot. This is an ordinary mid-grade plywood, suitable for a paint-grade project like a pin cab. The lock miter routing is very clean on the horizontal side, which is cut parallel to the grain. The vertical side is cut across the grain, which is harder on the veneer, but still comes out pretty clean, especially if we pre-score the veneer. I left a section at the top un-scored to show the difference that pre-scoring can make. This one actually isn't bad even in the un-scored section - I've seen much worse chipping in other projects. It probably helped that I routed this one in two passes.



Another view. There's a little blow-out (chipped veneer) at the end on the horizontally routed side - this is the trailing end where the router bit came out. I'll just sand it, but you could probably prevent it entirely by pressing a sacrificial board up against this end as you run the board through the router. On the vertical side, you can see that the lock tongue didn't survive the routing fully intact. It should be a perfect mirror image of the horizontally routed side, but it's quite stunted compared to the other side. And what's left is so delicate that it will easily flake off if you're not really gentle with it. The reason this comes out so poorly is that we're trying to make a very thin strip out of the layered plies, and that just

doesn't work. The horizontal side works better because it leaves basically one whole ply intact for the tongue. (And it's unavoidable that we have to route the vertical side into the plies like this - it's inherent in how these bits work.) There's enough structure remaining that the join will still work, and it'll still self-align like it should, but we do lose some strength from the missing material.



Dry fit of the two pieces above. The lock miter is tight enough to hold the corner together with friction for a test fit, which makes it really easy to do the final assembly.

After pointing out all those flaws, I should clarify that the end result still usually *looks* nice once the corner is assembled. The flaws are mostly on the inside, where the glue goes, and where you can't see them in a finished cab. Most of my attempts have yielded nice sharp corners and unblemished outer veneer.

Will you get decent results with the plywood you're using? I can't say for sure, given the mixed experiences I've seen on the Web. The lock miter clearly works with some plywood, but maybe not all plywood. It's probably a function of the particular material and/or the particular router bit. I think the only way to find out if it'll work for your batch of plywood is to test it. One nice thing about the elaborate setup process is that it can also serve as a fitness test for the plywood stock, since you have to make a bunch of test cuts and construct at least one test corner. You

should be able to tell from the test cuts whether or not you'll get a clean enough result. If you're not happy with the way things look during setup, you can always fall back on the plywood-friendly technique instead.

Equipment

Lock miter router bit: These are available from several manufacturers. They all use the same basic idea of the mirror-image shape and the horizontal/vertical routing process. I use an inexpensive no-brand bit I found on Amazon, and it does an acceptable job.

Note that there are also two-piece lock miter bit sets that use complementary bits for each side of the corner rather than a mirror-image pattern. Those require a different setup not covered in this section.

Setup jig: Optional, and in my opinion, not all that helpful. If your bit comes with one, go ahead and use it, but I wouldn't go out of my way to buy one separately from the bit. It's too hard to make sure that the jig exactly matches the bit if you have to buy them separately, since different bits have subtly different shapes.

The point of the setup jig is to help set the initial height of the bit without having to check it against the board you're cutting. The thing that makes this of dubious utility is that you still have to go through the test-and-adjust procedure described later in this section, since it's so important to calibrate the bit height to the actual plywood you're using. A setup jig helps with the initial height estimate, but that's not the time-consuming part, so it doesn't really save you that much.

Router and router table: If you already have a hand router, you can buy a bench-top router table and use it with your hand router. Most of the the bench-top tables are compatible with many routers from many brands, so you can generally mix and match brands. I use a relatively inexpensive table from Skil, which works pretty well. It's not a high-end piece of precision equipment, but it's been good enough for the pin cab joinery I've attempted, including lock mitering.

I'm afraid I don't know of any way to use lock miter bits with a hand router alone - I think you really need a table for this job.

Router table fence micro-adjuster: Optional but really helpful. Provides a way to adjust the fence position in tiny fractions of an inch, to help get the alignment perfect. This is something you can make yourself as a simple DIY project, as described in Appendix 12, Lock Miter I: The Plywood-Friendly Way.

Vertical sled: Optional but really helpful, especially when lock-mitering for a project that involves large pieces (such as a pin cab). This is another DIY project discussed below.

Router bit setup procedure

Important: don't set up your lock miter bit until you're ready to route **all** of the pieces that you want to route with it. The setup is the hard part, so you want to be able to set it up once and do all of your routing in one go.

The lock miter bit has to be set up so that the height and depth of the cut are exactly matched to your plywood stock. The only way I've found to do this is to do a series of test cuts, and make small adjustments based on how well the test pieces fit together.

The bit height and fence depth settings are independent, so you can get one dialed in

first, then do the other one. The procedure that seems to work best is to get the height adjusted perfectly first, then set the depth.

The setup all hinges on the symmetry of the cutting pattern. The goal is to get the pattern perfectly centered on each board. When the centers are aligned, the corners are aligned. Remember that each corner will consist of one piece that runs through the router horizontally, and one piece that runs through vertically. The router bit has to be set so that its height above the table centers the bit relative to the horizontal cut, and the fence has to be set so that the bit is centered for the vertical cut.

For safety, always unplug your router before making adjustments to the bit height or fence position.

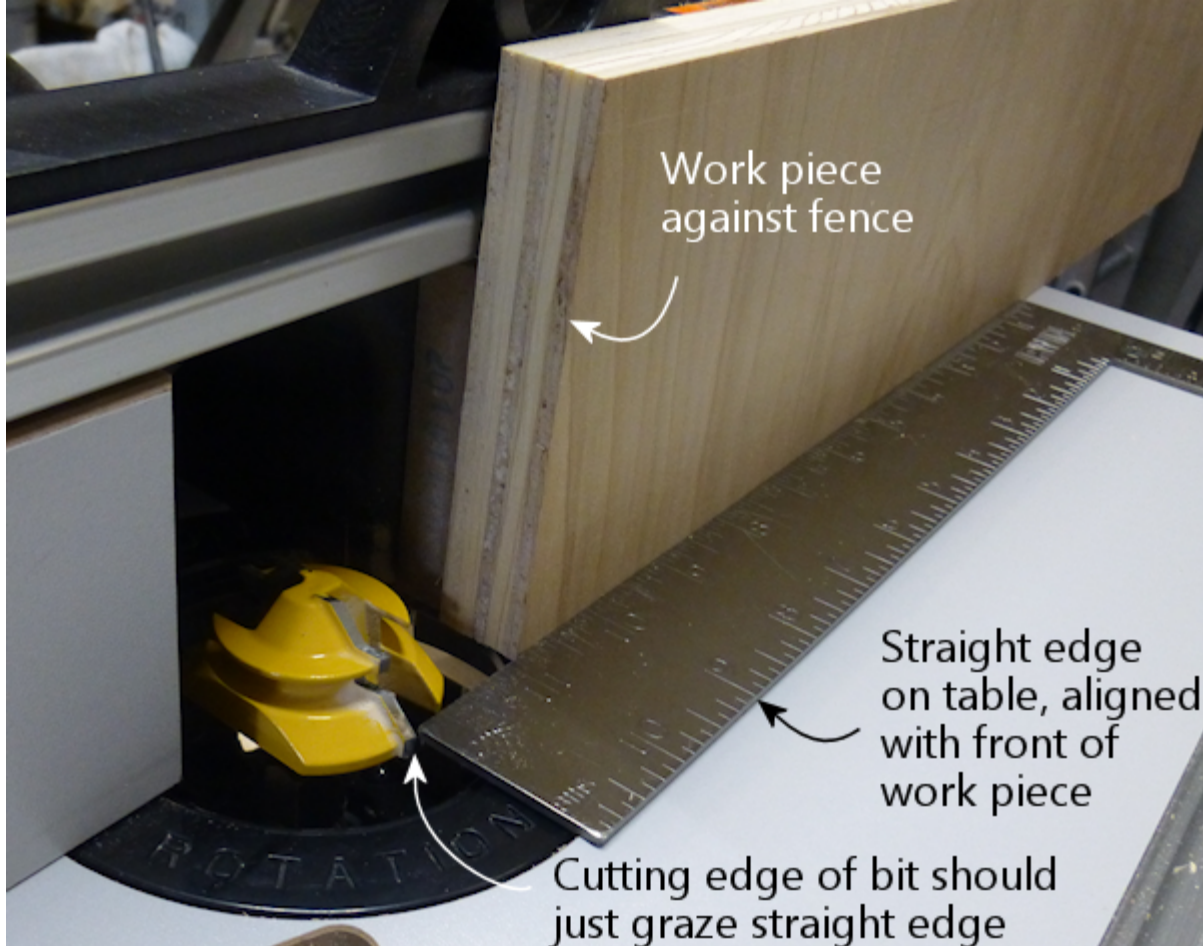
Step 1: Cut a piece of plywood to use for setup and testing, using the same stock you're using for the lock miter corners. The exact size isn't too important, but something like 5" x 18" should work.

Step 2: Put the test piece on the router table next to the bit. By eye, adjust the bit height so that the board is centered within the bit's slanted cutting area.



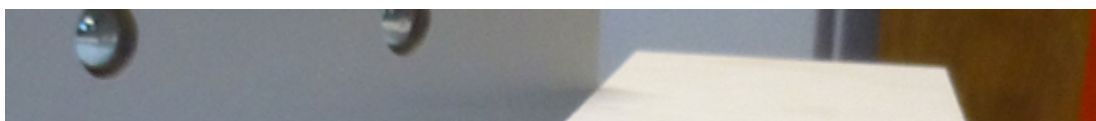
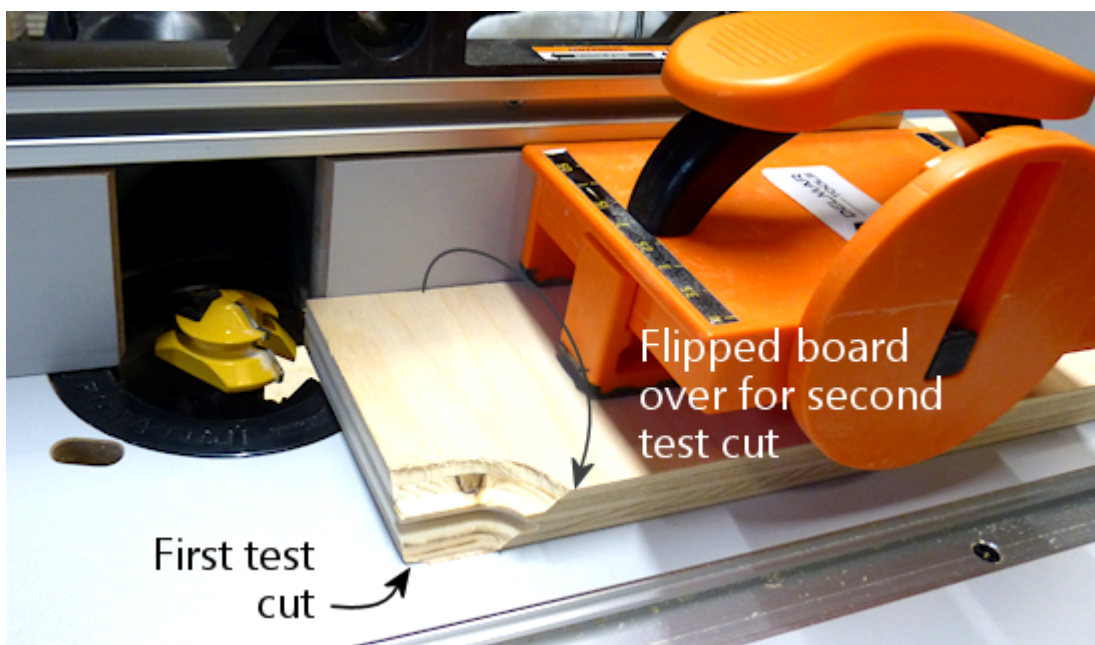
Most of the Youtube videos on the lock miter want you mark a center-line on the board, and align that with the center of the bit. If you like that idea, go with it. I find it easier to judge by the overall height, especially with 3/4" plywood and a 3/4" bit, since the cutting area of the bit is just a hair taller than the board. It's mostly just a matter of getting the bit lined up so that the board is completely within the cutting area. If you're using a larger bit or a thinner board, it might be easier to work with center marks, but even then I'm not sure. At any rate, it doesn't have to be perfect at this point, since we'll micro-align it based on a test cut shortly.

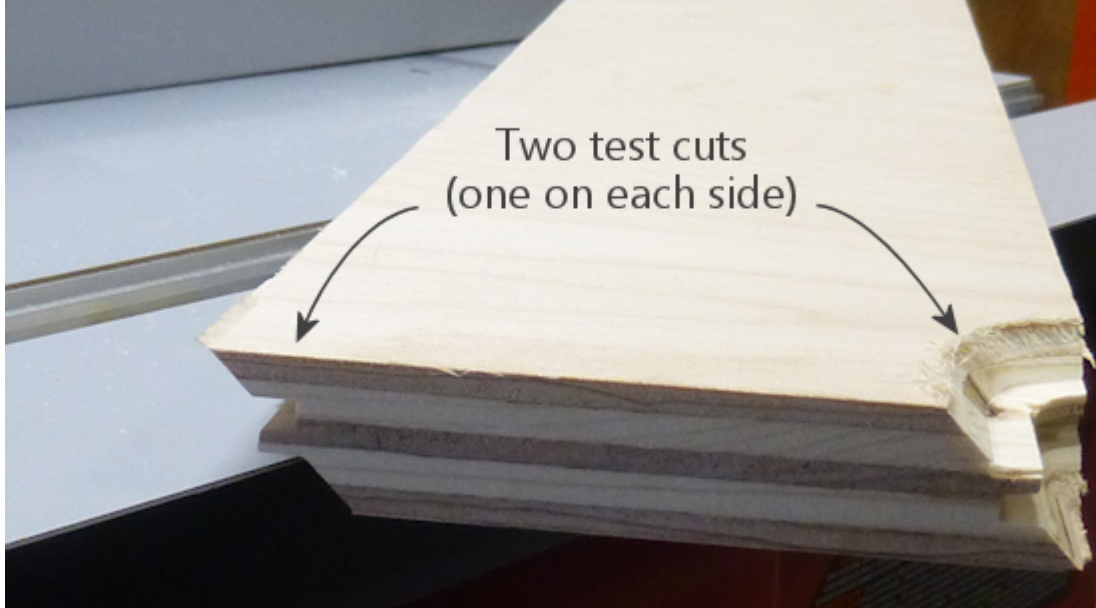
Step 3: Set the initial fence position. As with the initial bit height, this is only an approximate starting point, so you don't have to spend a ton of time here. Rotate the bit so that the cutting edge is pointing straight out at you (make sure it's unplugged first!). Put a metal straight-edge on top of the board and against the fence, and slide it across until it crosses in front of the bit. The goal is to align the fence so that the ruler just barely touches the cutting edge.



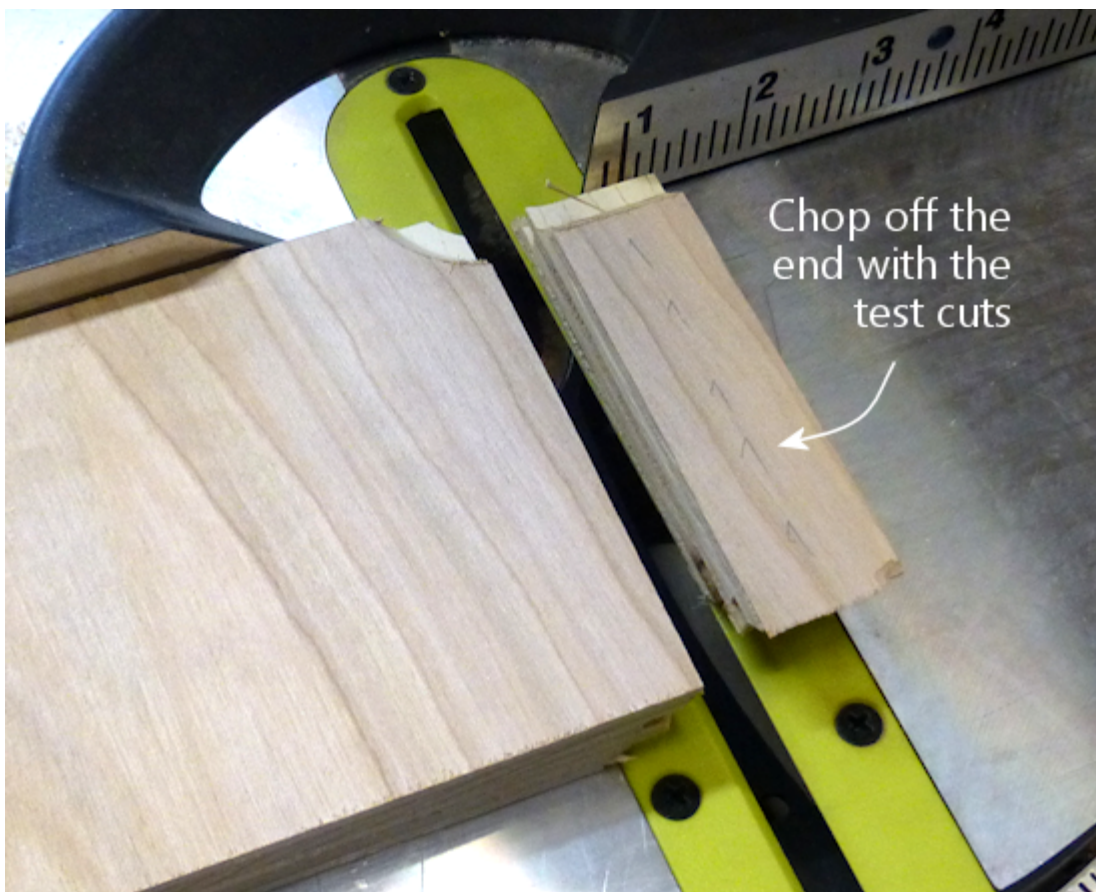
Note that most of the Youtube tutorials use this method to set the final fence position, but I've never been able to get it accurate enough this way. We'll fine-tune the fence later using a more precise method.

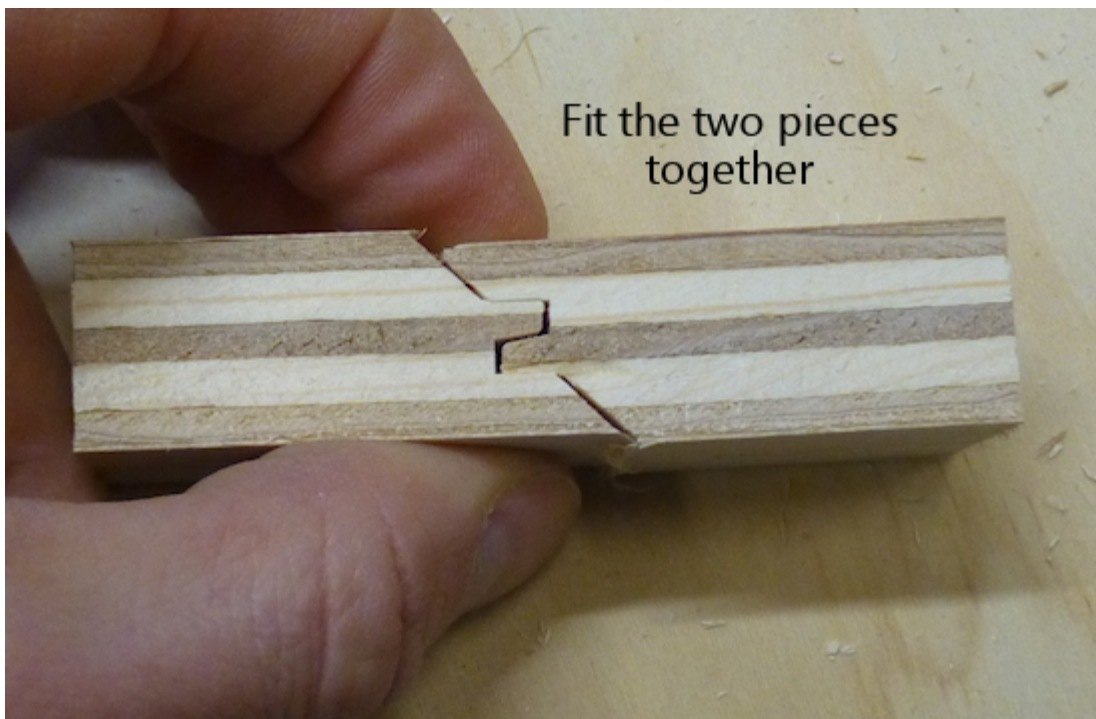
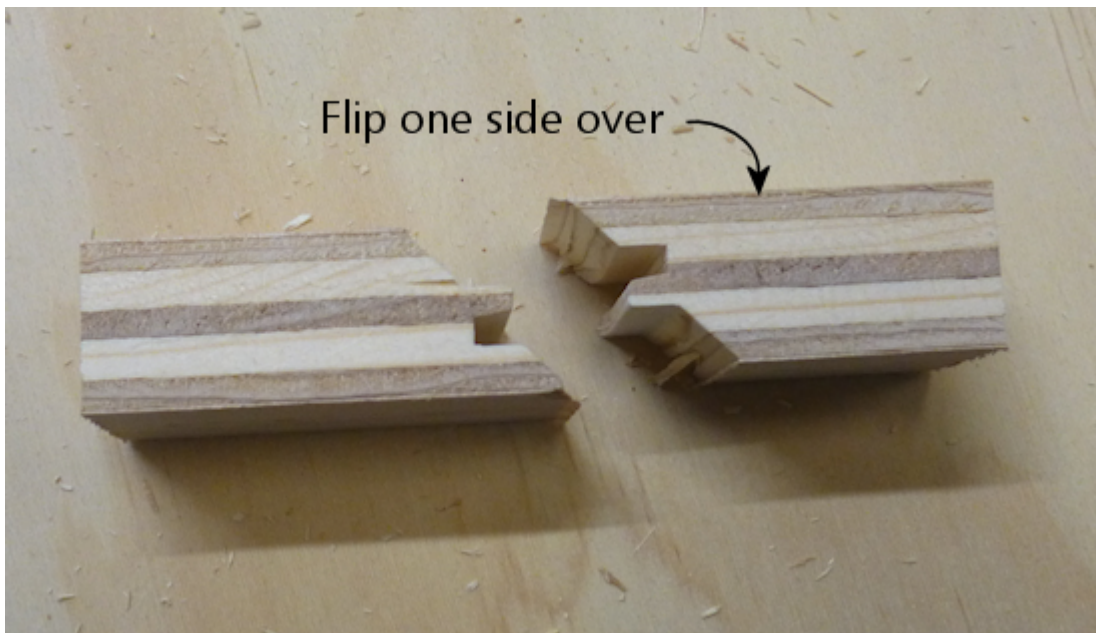
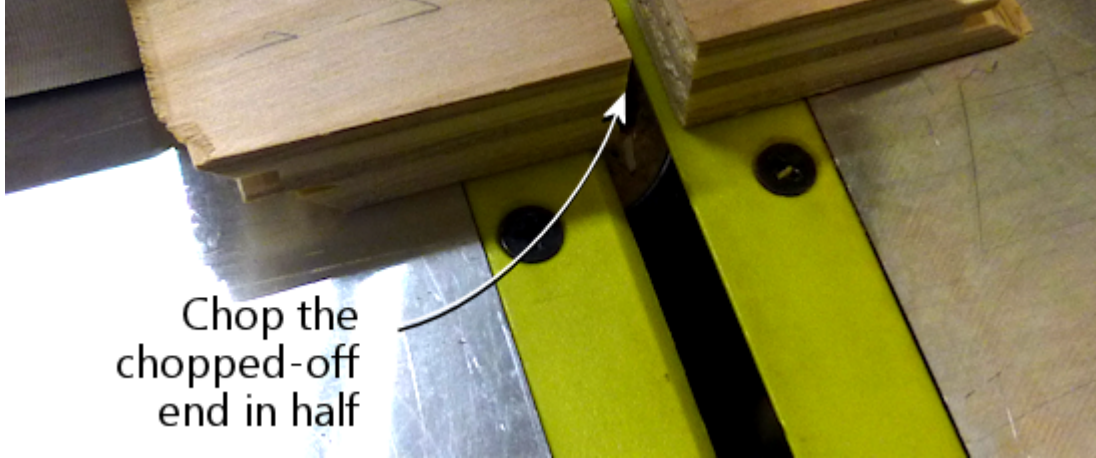
Step 4: Place the test board flat on the router table, against the fence. Turn on the router and feed the board in for just a short distance - about 2". Take it out, flip it over, and feed it in again from the other side, again for just 2" or so.





Step 5: Cut the short section you just routed off the end of the board, then cut that strip in half. Flip one half over and fit the routed sections together to test the fit.





Step 6: The goal is for the two pieces to be aligned perfectly. The top and bottom surfaces of both halves should line up exactly, so that you practically can't even feel the seam when you run your finger over it. If by some miracle they're exactly aligned

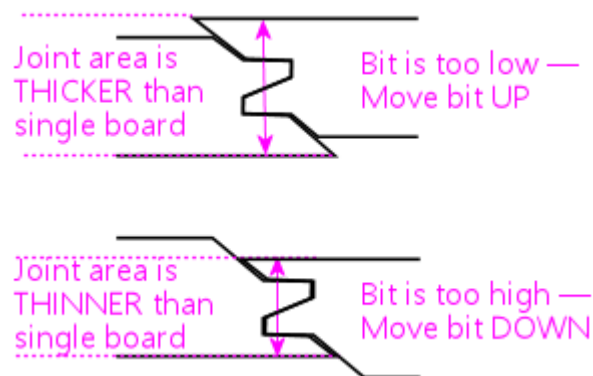
after that first test cut, you're already done setting the bit height! But they'll probably be a little off at this point, as in the test fit photo above.

To correct for the offset, you have to move the router bit up or down by half of the offset distance. So ideally, you need to know the exact numerical size of the offset. If you have digital calipers, measure the thickness across the joint section, then subtract the thickness of the board itself (as measured with the calipers) to get the offset distance.



If you don't have calipers, you can try using a ruler, or you can just make an estimate by eye. You'll converge on the magic spot more quickly if you can get a more precise reading, though. I can usually get it just about exact on the second try when I use calipers.

Step 7: Move the router bit up or down by **half** of the offset distance, according to the direction of the offset:



*Which way to adjust the router bit, based on the direction of the offset. The easiest way to tell is by looking at whether the joint section is **thicker** or **thinner***

than the individual boards.

Most routers have a micro-adjustment dial for the depth that lets you change the depth in tiny fractions of an inch; for example, my router's dial has ticks at 1/128". Refer to your router manual if you're not sure how to use that. It helps a lot in this step to be able to control exactly how much you're changing the depth on each iteration.

Step 8: Repeat the whole test, and check the new vertical offset. If it's still off, measure it again and adjust the bit height by half of the offset distance, using the same up-or-down rule as before. Keep repeating until the two test pieces align exactly.

Try to get the alignment practically perfect before declaring victory and moving on to the next step. Any error at this step will manifest as double the error in the cabinet width and/or length, which could affect the fit of your lockbar, TV, or other parts.

Step 9: Once the height is perfect, we need to adjust the fence position to get the depth perfect. The procedure is the same, but this time you do vertical cuts instead of horizontal cuts.



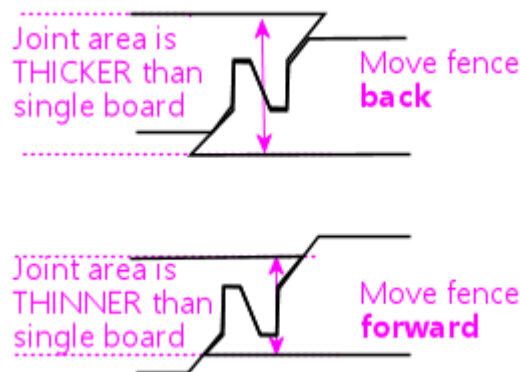
Preparing to make the vertical test cut. Hold the test board vertically against the fence while running it through the bit. It's helpful to use a featherboard to keep it pressed against the bit without placing your hands close to the bit (a push block would also work).





As with the horizontal test cuts, make two short (2" or so) cuts, cut off the ends with the routing, flip one piece over, and fit them together. The goal is to have the top and bottom surfaces perfectly aligned. If they're not aligned, we need to adjust the fence to make the cutting depth deeper or shallower.

In this case, if the joint section is too thick, move the fence back, away from the front of the table. As before, we move the fence by half of the offset distance.

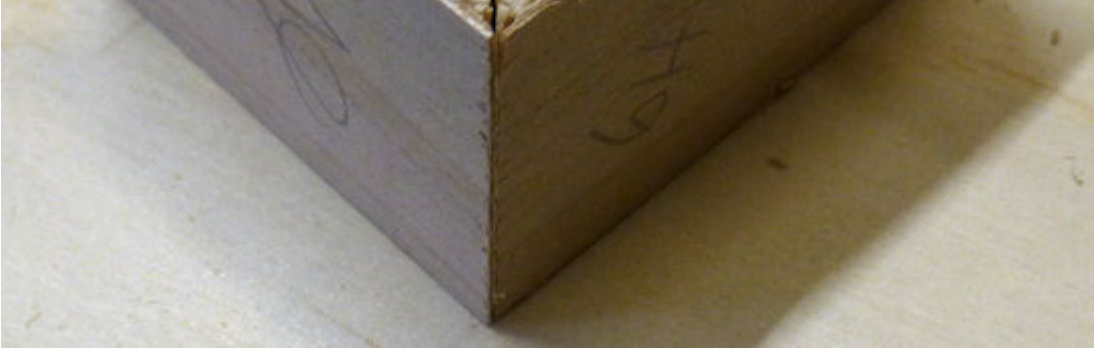


As in the previous step, repeat the test-and-adjust process until the two pieces are exactly aligned.

This is the step where the value of the Appendix 12, fence micro-adjuster mentioned earlier becomes obvious. Before I built the micro-adjuster, this step could be incredibly frustrating as I kept overshooting the magic center spot in one direction and then the other, trying to estimate tiny fractions of an inch by eye. The screw adjuster makes it a lot more controllable.

Step 10: The height and depth should now be dialed in. I'd do one final test run at this point, but this time, make an actual corner joint: using two scrap pieces, run one piece through horizontally, and run the other piece through vertically. Fit them together to make a corner. The result should have the seam exactly at the corner.





Remember that any deviation from the seam being exactly at the corner will throw off the assembled width and/or length of your cabinet by twice that amount, so don't settle for close-enough. Even if you can tolerate a little cosmetic imperfection in the corner placement, the size deviation after assembly could turn out to be a practical problem. It's worth trying to get it perfect at this stage.

If it looks good, you're ready to route your actual work pieces. Don't touch the router setup again until you're done routing all of the corners - you want to make sure the perfect alignment remains locked in until you're done.

How to route the corners

Before you start doing the routing, I have a couple of other pieces of advice to help improve your results, so you might want to read through the sections below before proceeding.

When you're ready to perform the routing, start by going around the corners and designating the horizontal and vertical routing sides. You need one horizontal side and one vertical side at every corner. It's up to you which is which, but you have to be sure that every corner uses complementary orientations for its two adjoining faces.

My scheme is:

- Front face is routed horizontally on both sides
- Back face is routed horizontally on both sides
- Left side is routed vertically on both ends
- Right side is routed vertically on both ends

I do it that way because I find it easier to handle the long side pieces in a vertical orientation by using a "sled" (see below).

To make sure that I don't get anything confused, I always take a pencil and mark each of the edges I'm going to route - "Lock Miter Here Horizontally This Face Down", "Lock Miter Here Vertically This Face To Fence". All of the routing must be done on the **inside** faces, so I mark the inside faces only. I like to include "Face Down" or "Face To Fence" so that I'm more likely to catch myself if I'm about to feed a board in the wrong way, since I'll be looking at a face-up marking telling me it needs to be face-down.

Once you have everything marked, checked, and double-checked, it's just a matter of running the edges through the router as marked. Again, always be sure to route with the inside face down (for horizontal cuts) or towards the fence (for vertical pieces).

After all the routing is done, assembling is just a matter of fitting the pieces together.

Lock-mitered corners tend to hold together pretty well just by friction, which makes it easy to do a dry fit to test that everything aligns properly.

Use a second layer to reduce chipping on the outer face

In order to make perfectly seamless corners, the lock miter bit has to cut right out to the edge of the plywood. This makes the edge so thin that it can easily chip during the cutting process.

The way to minimize chipping (and hopefully prevent it entirely) is to place a sacrificial board right behind the board you're cutting. The extra layer keeps the outer veneer from flexing as the bit hits it and helps keep it in one piece. Use scrap material, since it might get dinged just a little along the edge.

If you want, you can attach the extra top layer to the work piece using woodworker's tape, which is a thin, double-sticky tape designed for just this kind of job. I tried the tape a few times, and eventually decided that it was easier to skip it, and just keep the pieces together by hand. It helps a lot to use a featherboard on the router table fence to press down on the workpiece - that helps keep the two pieces pressed together even better than the tape does.



Using a second piece of plywood on top of the horizontal work piece, to prevent chipping on the outer veneer. The second piece should be aligned exactly with the fence side of the board you're cutting; you can fasten it to the main board with woodworking tape to keep it fixed in place throughout the cut. Note that I'm set up for a shallow "first pass" here, using MDF spacers on the fence, as described below.





Using a second piece of plywood to prevent outer veneer chipping on the vertical cut. I'm using a featherboard on the table to keep the two pieces pressed against the fence through the cut.

Make the cuts in two passes

Your router probably comes with advice saying that you should always make deep cuts with multiple passes. My router manual suggests going no more than about 1/4" deep on each pass.

Well, if you look at this bit, it's quite a lot more than 1/4" deep. Plus, it cuts a wide swath.

I've seen improved results by making each cut in two passes. The lock miter bit's geometry lets us make multiple passes as long as we do it by adjusting the fence depth (not the bit height - that must stay identical for all passes).

The problem is that it's such a pain to get the fence aligned perfectly that we don't want to touch it once it's set. But there's an easy way to adjust the routing depth without moving the fence: attach a little extra spacer in front of the fence. A thin piece of MDF - 1/4" to 3/8" thick - works great for this. Cut pieces roughly the same size as your fence halves, and attach it to the front of the fence with small pieces of woodworking tape. After completing the first pass on all pieces, remove the MDF spacers, and run the pieces through again directly against the fence.

Pre-score the inner veneer to reduce chipping

My lock miter bit tends to make a nice clean cut on cuts that are parallel to the grain. When making cross-grain cuts, though, the bit can make a rather bad mess of the inside veneer, by knocking out lots of little chips along the edge.

The obvious solution would be: don't do that! Unfortunately, you can't avoid cross-grain cuts in a full-sized pin cab, because the side walls are over 48" long. That

forces you to orient the side walls "the long way" when cutting up a 4x8 sheet of plywood, which means that the front and back edges of the side wall will necessarily be oriented perpendicular to the grain.

One way to deal with the inner veneer chipping mess is to just live with it. It can look ugly, but it only happens on the inside face, so it won't affect the exterior appearance. Plus, most of the length of the inner corners gets covered up with the corner bracing wedges, so most of it won't even be visible when you look inside the machine.

Another "deal with it" fix is to apply wood filler to cover up the chipped edge. If you do that, I'd wait until after assembling the cabinet, so that you don't accidentally fill in any of the miter cut.

I'd prefer to avoid the chipping in the first place, though. There's a technique that can help at least mitigate it. The idea is to pre-cut the veneer layer right along the line where the router bit operates. The router bit will still chip the veneer, but the chips should break off at the pre-cut line, so there should be no damage beyond that point.

To do this, before routing, draw a line on the **inside** face, parallel to the edge, in from the edge by slightly more than 3/4" (the thickness of the plywood). The lock miter cut has the same thickness as the plywood, so this will be just slightly inside of where the bit will touch the wood. Using an X-acto knife or sharp utility knife, score the plywood along that line, all the way down the edge, cutting all the way through the outer veneer layer (but just that deep). I'd use a metal straight edge, clamped to the work piece along the cut line, so that you can guide the knife by holding it against the straight edge.

Important: Only do this on the face that goes on the **inside** of the cabinet. Don't cut up the veneer on the outside - the whole point of the lock miter is to make that look pretty by avoiding any seams or cuts on the face.

Handling the work piece for the vertical cuts

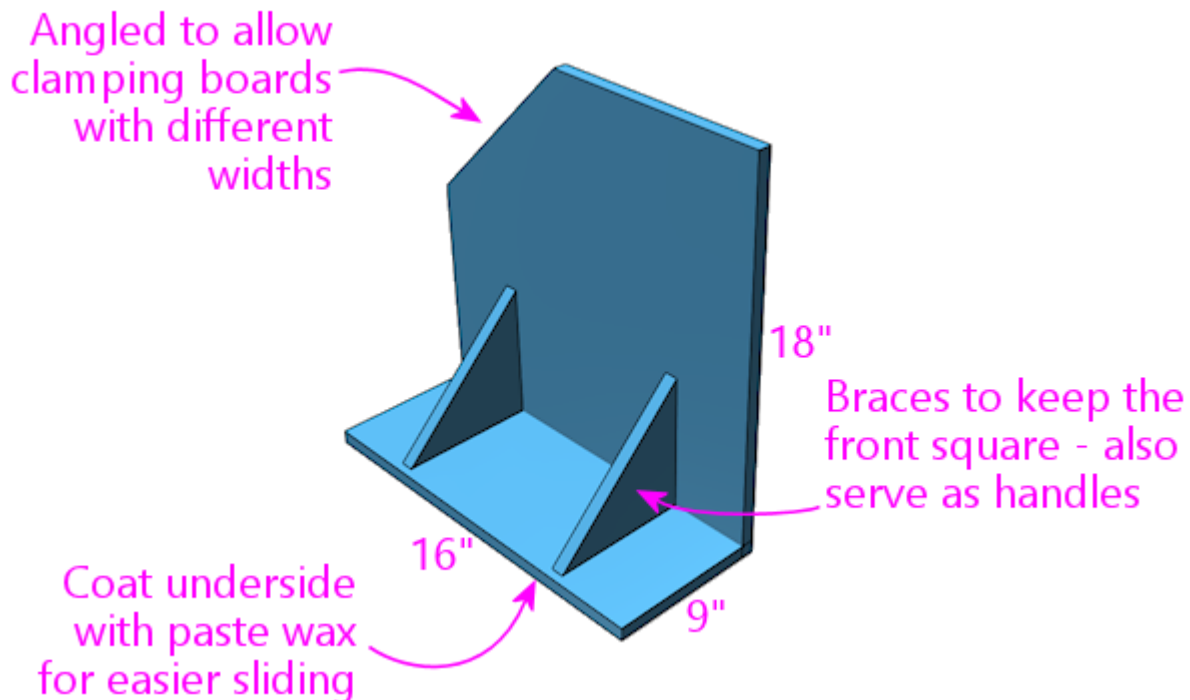
The design of the lock miter bit requires running half of the pieces through the bit vertically - standing them on end and holding them against the fence, rather than laying them on flat on the table. This is a challenge for tall pieces, because the fence on most router tables is only a few inches tall. It's difficult to keep a tall piece from wobbling, which can make the cut uneven and damage the edge.

The way I arrange the pieces, the side walls get the vertical treatment. So we have a 51" tall piece held vertically against a 4" tall fence - not exactly easy to manage. I haven't found any way to avoid this step, but I can at least suggest some extra tooling that helps a bit. The idea is to build a "sled" that holds the work piece in the awkward vertical position, so that you don't have to do that entirely by hand. You clamp the work piece to the sled, and then you slide the sled across the router table.

This "sled" is something that you can either buy or build yourself. This is a common enough woodworking problem that there are a couple of retail options available - search for "vertical router sled". But you might be better off building one, because the retail options I've seen aren't beefy enough to hold pieces as big as a pin cab side panel. It's actually a pretty easy project to build. I'll outline the design I used below.

My plan isn't especially clever, and there's no need to follow it exactly. All that's important is the basic shape and approximate size. The two main things to pay attention to are (1) making the corner joint square, so that the work piece is kept at

90° to the router table, and (2) making the vertical part tall enough to keep the work piece steady. The whole contraption needs to be solid enough that nothing sways or wobbles while you maneuver it through the router.



Everything is 3/4" plywood, including the triangular braces. I routed shallow 3/4"-wide dados for the braces and glued them. The seam between the front and bottom pieces is a lock miter join made with my lock miter bit, and it's also glued. The exact dimensions aren't important, but here's why I chose the ones I did. The width is enough that I can use an F-clamp on each side with the wide end (23½") of a cabinet side wall. The height is more or less arbitrary, but this is enough contact area to keep a 50"-long cab side wall steady once it's clamped down. The depth is about the same as the depth of my router table to the fence - anything longer than that would just overhang in front and add useless weight.

Note that the join between the front and bottom pieces is a good candidate for a lock miter, since you want this corner to be as square as possible. You'll still need bracing, and the bracing alone should be sufficient to keep everything square, so other joins are fine too. But it makes a good practice project if you want to try out the lock miter procedure before applying it to your pin cab.

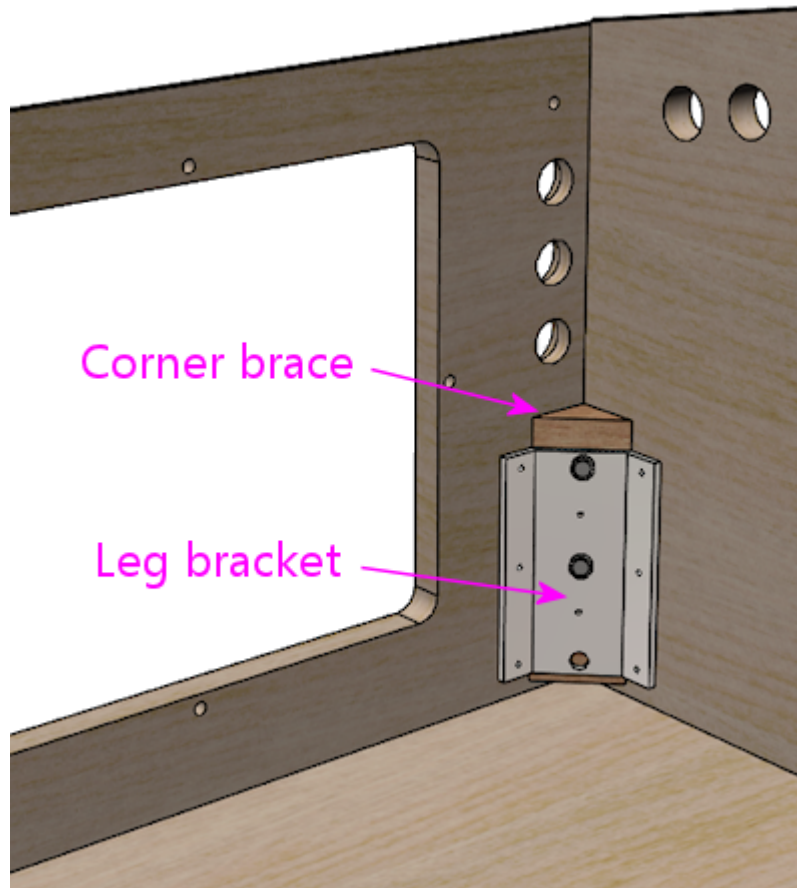
Summary of recommendations

- Build a simple micro-adjuster for your router fence, to make precise fence positioning easier
- Set up the router bit first by height, then by fence position, using a series of test cuts and compensating adjustments to get the bit perfectly centered by height and fence depth
- After adjusting the bit and fence positions, test-build one corner using scrap material to verify that everything is aligned perfectly
- Aim for perfect corner alignment, because the error in each corner will result in twice that error in the overall cabinet width, which could affect the fit for the lockbar and TV (plus, perfect alignment will make the corners look perfect)

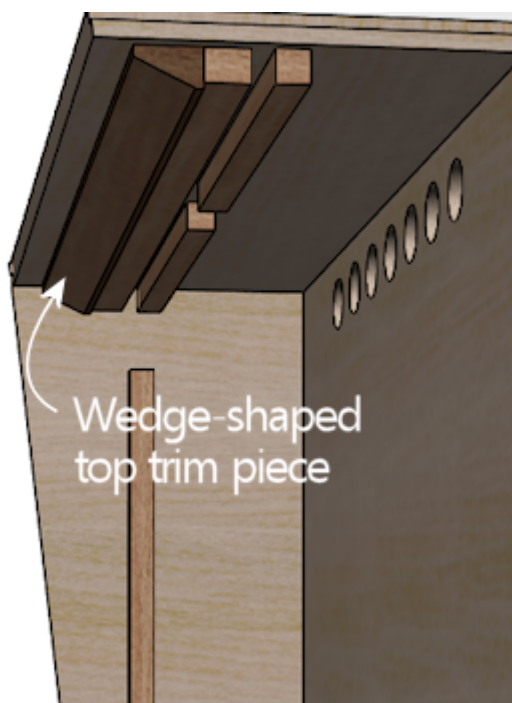
- Do the routing in two passes, with the first pass 1/4" to 3/8" shallower than the final pass; make the shallower first pass by using a temporary spacer in front of the fence
- Every time you run a board through the router, press another (sacrificial) board firmly against it on the outside face (the face away from the router bit) to prevent chipping on the outer veneer
- Before making a cut on an edge that's perpendicular to the grain, score the inside veneer with an X-Acto knife, just beyond the miter cut zone (so just slightly more than 3/4" from the edge), to reduce chipping on the inner veneer
- When making the vertical cuts, use a "sled" to keep the work piece square against the fence

Appendix 14. How to Make Corner Braces (and other wood prism shapes)

If you build a pin cab from scratch, you have to cut two types of wooden "wedge" shapes - or to be more precise, triangular prisms. One is for the corner braces that sit under the leg bolt brackets:



The other is the cosmetic trim piece at the top of the backbox:



When I first tried to make these for myself, I found them to be surprisingly

challenging. So I thought I'd offer some suggestions from what I've learned.

Pre-fabricated options

The simple triangular corner brace is the same shape as a type of trim molding called a **chamfer strip**. Unfortunately, it's not easy to buy such a thing. I've looked but haven't been able to find any retail sources. Apparently there's not enough demand that anyone wants to sell them.

The backbox trim piece is (surprisingly) easier to find. It roughly matches the shape of a common piece of floor trim called a **3/4" reducer molding**. You might be able to find these at your local Home Depot or Lowe's, and they're also available online. The floor trim usually has a tab that sticks out in back, which you'll have to cut off, but that should be easier than fabricating the whole thing. You can probably even do that with hand tools, such as a utility knife, hand saw, or jigsaw.

How to make them yourself

When I first tried to make these pieces myself, I realized that it wasn't going to be as easy as feeding a 2x2 into a table saw. In fact, the more I thought about it, the harder it seemed. So I consulted the Internet. Which didn't actually help a lot; I found a lot more questions about the subject than useful answers. But I did find a few interesting leads. Here's what I came up with:

- Most people asking about how to make pieces like this are trying to do it with a table saw. That's what I intended to use as well. And it *is* doable with a table saw - it's just not nearly as straightforward as I would have thought. Most of the rest of this section explains the table saw technique that I came up with, which worked well enough for me that I'll at least explain it. I'm not sure I should actually recommend it, because the main thing I learned from my Internet research is that this is a particularly risky kind of cut to make with a table saw. But I came up with some things that I think make it safer, so I'll share my approach so that you can make up your own mind about whether or not to use it.
- I found several comments suggesting that a band saw (with a table and fence) is the safest way to make this cut. I don't have a band saw myself, so I can't weigh in with any personal experience, but this does seem to be widely considered the best and safest tool for ripping narrow boards. Band saws don't tend to cause "kickback", which is what makes narrow ripping with table saws so dangerous.

If you have a band saw and want to give this a try, the technique should be straightforward. Set the table to the desired cut angle, set the fence to the correct depth, and feed the board through length-wise. See the sections below on the corner braces and backbox trim for the specific cut angles and dimensions.

- One person (making corner braces for a pinball machine, no less!) reported that he cut a 2x2 in half diagonally with a hand-held oscillating saw. I can't imagine doing this myself - my hand just isn't steady enough. The corner braces don't have to look pretty, but the parts that go under the leg brackets have to be fairly precise if they're going to fit well, and I just can't imagine making a straight enough cut with a hand-held tool. But maybe you can make this work if you're really careful.
- I was partially successful making these with a track saw. The trick is to firmly

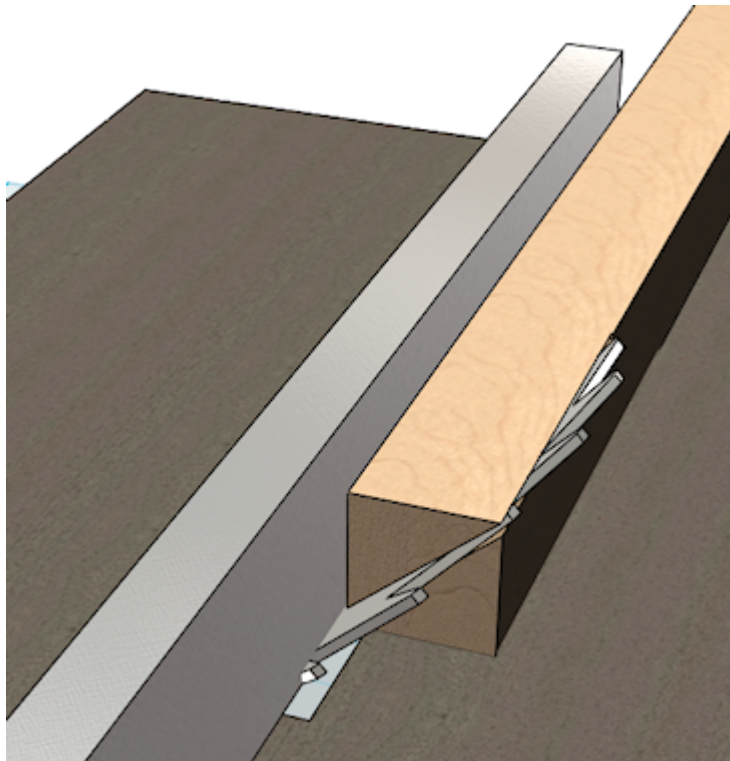
wedge the work piece (the 2x2 or 1x2) between scrap wood of the same height as the work piece. That forms a platform for the track to sit on (which is important because the track is much wider than a 2x2), and it locks the work piece in place throughout the cut. But I found it difficult to get consistent results this way. I also found that the depth of the cut was challenging for my track saw.

How to make them with a table saw

Warning: Table saws are dangerous tools. Please don't attempt any of this unless you're experienced enough with your equipment that you can evaluate the safety of what I'm suggesting. I'm a newbie at this stuff myself - don't take any of this as expert-approved. (By the same token, if you know a better way to accomplish these tasks, I'd love to hear from you.) If anything seems off or you're just uncomfortable with anything I suggest, trust your instincts and find another way that you're happier with.

Extra Warning: The big risk with making beveled cuts with a table saw is "kickback", which means that the blade grabs the work piece and throws it back at you at high speed. This can cause severe injuries. Your saw's owner's manual will have safety advice on how to position your body during a cut to reduce your chances of getting hit if kickback occurs. Please read that section and follow it carefully. (My table saw's owner's manual is one of the lengthiest of any tools I own, but I found it worth the effort to read it all.) Kickback is always a risk, but it seems especially likely with beveled cuts, so be extra careful.

Let me start with what *not* to do - which happens to be the thing that's most obvious, to me at least. Don't just take a 2x2 and try to feed it into the saw directly.

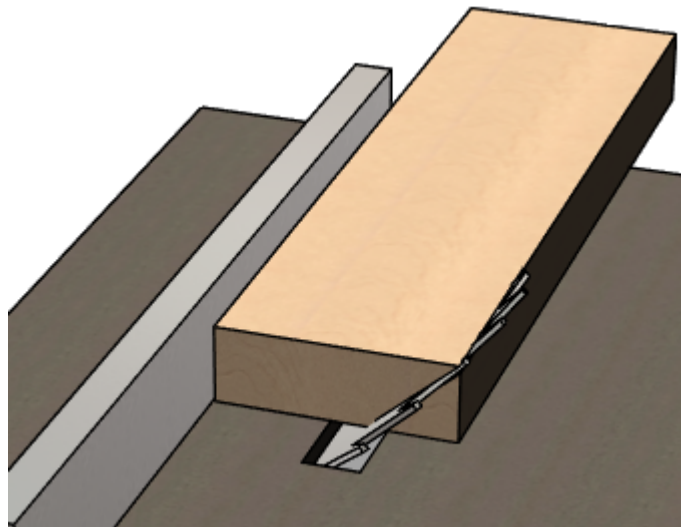


Don't do this. Ripping a 2x2 in half diagonally with a table saw, with the blade set to a 45° bevel cut. This is dangerous for several reasons.

Woodworkers call it "ripping" the board when you're cutting it down its length like this. Table saws are great for ripping sheets of plywood. They've even good for

ripping narrow boards like this, *if* the blade is oriented straight up. But a diagonal "bevel" cut like this is dangerous with a narrow board. The big problem is that the top half of the board coming out of the saw blade isn't supported by anything, so it'll tend to press down against the blade, which can potentially turn it into a projectile. It's also tricky to control a narrow board like this, even with a regular vertical cut, because there's so little room to push the board through between the fence and the blade. For a 90° cut, you can solve the second problem with a "tunnel" push block, like a Microjig Grr-ripper. But that doesn't help with the top-half support problem when cutting at an angle. In fact, I think it would actually make the top-half support problem worse, because the push block applies additional downward pressure on the unsupported half.

In my Internet searching, the best advice I found was to *not* try this with a 2x2 at all, but instead to use a wider board, and cut off just a corner. Let's see how this looks with a 2x6 in place of a 2x2 in the original setup:



Better: Use a 2x6 instead of a 2x2 to rip a wedge-shaped piece off one side.

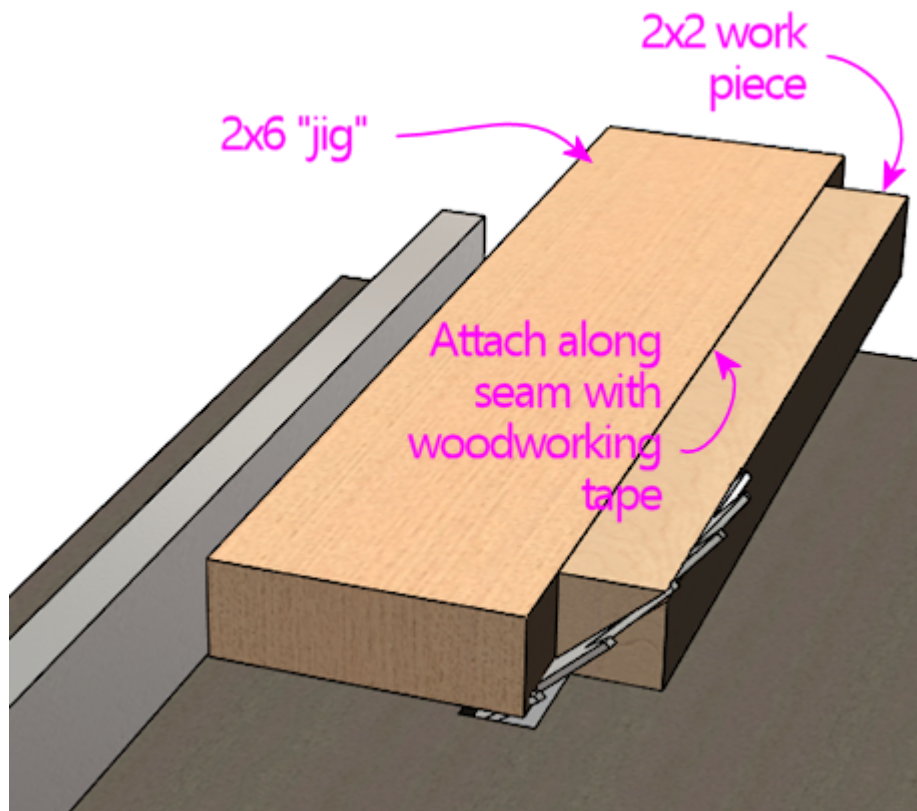
A 2x6 is the same thickness as a 2x2, so it yields the same result we were trying to get from the 2x2, but this seems a lot safer. Unlike with the 2x2, the "top" portion remains well supported throughout the cut - it's not going to tip over onto the blade. The wider board is also easier to control, and you have a lot more room between the fence and the blade to guide the board. The gap between the fence and blade is still tight enough that you need a push stick or push block, but at least it's not insanely scary this time. What's more, this setup lets you work with your saw's blade guard and anti-kickback pawls installed, which is a major safety improvement.

Note how the saw blade is oriented relative to the fence in the diagram above: the blade is tilted **away from the fence**. This is widely recommended as the safer way to orient the saw for a bevel cut, because this geometry is less likely to trap the work piece between the fence and the blade. I think this is true on any table saw, but just in case your saw has some special design that makes it different, please read the relevant section in your saw's owner's manual to make sure that it agrees. You should of course also check it for any other safety advice it has about beveled cuts.

The downside of using a 2x2 to make a 1" wide strip is that you end up with a lot of wasted material. You could get one more wedge shape out of this by flipping the leftover portion of board over and making a regular 90° straight-up cut just inside the newly beveled portion, but even then you have 2/3 of the board left over. I don't think you want to attempt a third cut, since the remainder is getting back to the point of being dangerously narrow.

This got me thinking about how we could do the same thing without using such a wide board. I think using a wider board like a 2x6 is probably the safest option overall, so maybe you should just stop here and do that, but the alternative I came up with worked pretty well for me. I built a very simple jig that holds a narrow board at a set distance from the fence. You move both boards through the saw at the same time, so that the wider board steers the narrow board through the cut. We're effectively making a shape like the 2x6 out of two boards, but we're only cutting up the narrow board, so there's less waste.

The simplest version of this jig is just a 2x6 and some woodworking tape. Take the 2x6, and attach the narrow board (a 2x2 or even a 1x2) to one edge, making sure that the bottom of the work piece is flush with the bottom of the 2x6. Attach it with "woodworking tape", which is a thin double-sticky tape made for just this kind of temporary attachment while you're working on a piece of material. Search for "woodworking tape" on Amazon for numerous options.



As with cutting a 2x6 directly, you should be able to use your saw's blade guard and anti-kickback pawls with this setup. Be sure that the work piece is securely attached to the wider board.

Instead of a 2x6, you could use two pieces of 3/4" plywood cut, stacked one atop the other and glued together, which gives you the same thickness as a 2-by. I'd use glue, not nails or screws, so that there's nothing metal for the blade to hit if you should ever accidentally cut through part of the jig.

It's important that the sides of the jig are straight and parallel. There are several easy ways to do this; do a Web search for "jointing with a table saw" for videos showing techniques. The basic idea is that you need another board that already has one straight edge that you can use as a reference against the fence, and then you use that board to guide the board that you want straightened (the jig, in this case) through the saw, to take off just enough material from the rough edge to make it perfectly straight. After you've "jointed" one edge of the jig this way, you can run the jig through the saw again, holding the newly straight edge against the fence, to

make the other edge perfectly parallel to it.

It would be possible to elaborate on this jig with something that holds the narrower board in place without tape, but I'll leave that to your ingenuity.

Corner braces

Here's the target size for the corner braces:



Unfortunately, this is *not* the size you get when you cut a 2x2 exactly in half diagonally.

To get the right size, you have to cut the 2x2 a little off-center. Exactly how far off-center depends on the thickness of your saw blade, so the easiest thing to do is probably to make a series of test cuts, and measure and adjust until you hit the right size.

If you want to test the fit against an actual leg bracket, grab a couple of small pieces of scrap wood, make a corner out of them, and screw a leg bolt bracket to the inside of the corner. Then slip the test piece under the bracket to see if it fits. When the size is right, the test piece should fit snugly.

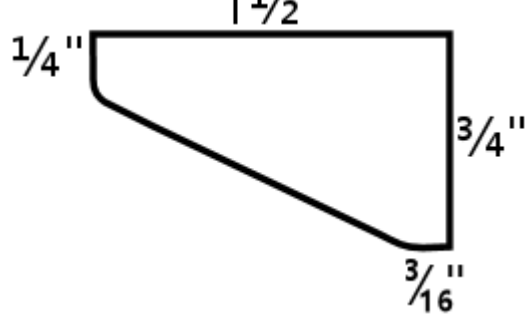
The diagonal angle is 45°, so set your saw blade at a 45° bevel tilt.

Lengths: The most critical function of the braces is to fill the gap under the leg bolt brackets. The brackets are about 5½ inches long, so you can satisfy this function by making all four corner braces about 6" long. Alternatively, you can make them a bit longer, so that they provide additional reinforcement along a greater section of the corner seams:

- At the front, the braces can extend from the cabinet floor to the top of the brackets, which amounts to about 8½". You shouldn't make them extend above the top of the brackets, since they might get in the way of the plunger and front-panel buttons.
- At the back, the braces can extend all the way from the cabinet floor to the top, about 21½".

Backbox top trim

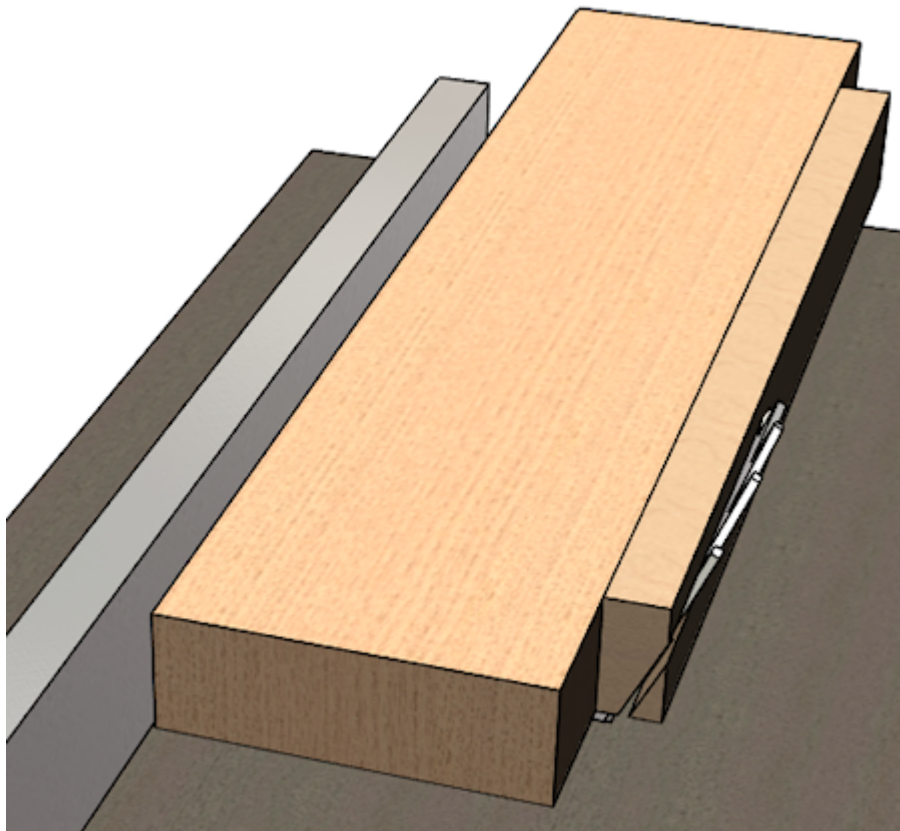
The backbox top trim piece has this profile:



This piece is purely cosmetic, so you don't have to hit those dimensions perfectly. Those dimensions are just what the WPC machines use. I don't think anyone would notice the slightest difference if you had a bit more or a bit less than the 1/4" lip at the front.

You can accomplish this shape by cutting a 1x2 with the blade set at a 20.5° angle. The 1x2 should be oriented the "tall way", and set up the fence offset so that the bottom edge is cut to 1/4" thick. (Approximately - again, it doesn't really matter that hit that size exactly.)

As with the corner braces, the easiest way to get the fence position set up is to make a test cut, and adjust the fence based on the results.



Appendix 15. A DIY shaker motor plan

This section provides instructions for building a pinball shaker motor from scratch, using fairly common and inexpensive parts. The only parts that you have to fabricate yourself are some simple plywood pieces that require only straight cuts and some drilling.

My design is hardly the first or only plan out there - see Chapter 61, DIY Designs in Chapter 61, Shaker motors for a list of other plans published on various forum sites. The reason I developed yet another design is that the existing plans all have something that makes them a little bit challenging to implement, either because they call for an obscure part or because they require some difficult metal fabrication steps. My aim with this plan is to build it entirely with easy-to-find parts and minimal fabrication work, and still match the performance of the commercial shaker units. I also tried to keep the finished price to a minimum. Excluding the motor, the parts in my plan run to about \$30; with a motor, the price should come in at about \$50 to \$60.

At the moment (2022), there's little cost advantage to building a shaker yourself, since you can buy a complete, ready-to-use unit from Pinball Life for about \$100. Even so, I like knowing that there's an easy DIY plan available, in case the retail shaker units become unavailable in the future. The Pinball Life units will eventually sell out, and who knows if they'll restock when that happens. Pinball parts like this historically come and go. I expect there will again come a time when you can't buy these so easily.

Cost and availability aside, a DIY shaker has the advantage that it gives you more control over the effect than you get with an off-the-shelf unit. The retail units (the current ones, anyway) don't provide any way to adjust the amount of weight or the geometry. You can still gain some control over the force of the shaking effect by varying the motor speed, but that also affects the "tonal" quality of the effect, so it's not a pure strength control. With the DIY plan laid out in this section, it's easy to fine-tune the total amount of mass in the counterweights, so you can adjust weight and motor speed independently to optimize the effect to your liking.

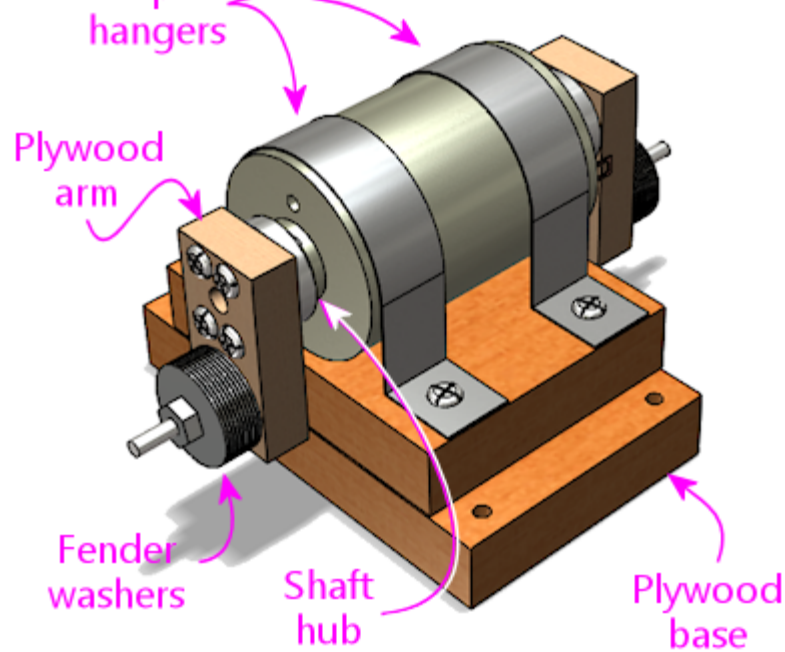
Design overview

There are really just three puzzles you have to solve to design a shaker motor: what to use for the counterweights, how to attach the counterweights to the motor shaft, and how to mount the motor in the pinball machine cabinet.

This design uses fender washers as the counterweights. They're easy to attach to almost anything, since their whole purpose is to fit over a screw, and they're fairly heavy, being made of steel. It only takes about ten 1" diameter washers to get the 50g of weight we're after in one counterweight.

To attach the counterweights to the motor shaft, we use plywood lever arms and shaft hubs (more on those below).

The motor is mounted to the cabinet via a plywood base (which also serves to elevate the motor enough to make room for the weights to spin without hitting the floor), and it's secured to the plywood base with pipe hangers (or U-bolts, if you prefer). Pipe hangers are common plumbing hardware for attaching metal pipes to walls and ceilings, which makes them the right shape for bolting down a cylindrical motor body.



The plywood pieces are the only things you have to fabricate. I figure that most pin cab builders will have some plywood on hand anyway, along with a drill and some kind of saw, so plywood fabrication should be pretty approachable. Otherwise, it's all off-the-shelf parts, and most of those are basic hardware items that you can find at the likes of Home Depot.

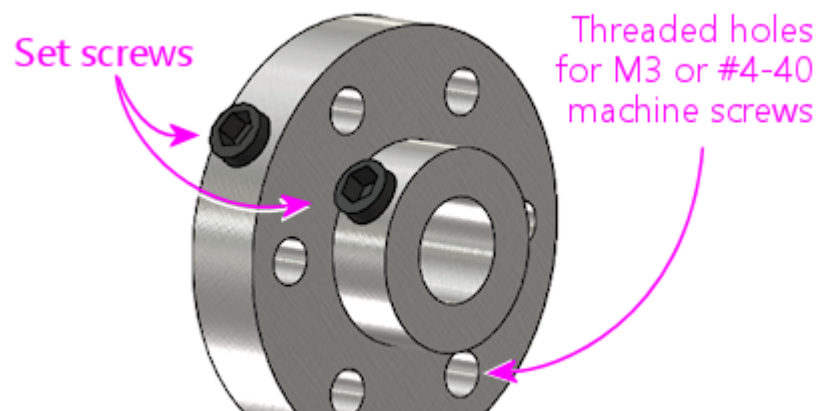
Pre-fab motor brackets


Many of the DC motors that you can buy on eBay and Amazon have their own ready-made mounting brackets available. If the motor you find has a matching bracket, I'd go ahead and buy it, since that should do a good job of securing one end of the motor to the base. You can use that in place of one of the pipe hangers. Most of these special brackets only attach at one end of the motor, so you'll probably still need to include one pipe hanger, to secure the end opposite the bracket. For a shaker, it's critical to immobilize both ends of the motor body.

Shaft hubs

Most of the parts in this plan are ordinary hardware store items. The only exceptions are the motor itself, and the "shaft hubs". Shaft hubs are fairly easy to find online - they're used a lot in hobby robotics, mostly to attach wheels to motors. You can buy them at Pololu, Amazon, eBay, or Aliexpress. They currently run about \$10 a pair.

Look for something like this:





The key features are a set-screw that lets you fasten the hub to the motor shaft, and threaded holes for machine screws in the larger disk.

Pick out your motor before buying shaft hubs, because the hubs have to match the motor shaft size. You need a hub with a center bore the same size as the motor shaft diameter. Shaft hubs are available with center bores of 3mm, 4mm, 5mm, 6mm, 8mm, and 1/4", which are common sizes for DC motors shafts. You should be able to find the motor's shaft size listed in specs on the seller's site.

The holes around the perimeter of the outer disk should be threaded for machine screws. These are usually threaded for either metric M3 screws or US #4-40 screws. Pololu makes both varieties. When you buy the hubs, take note of the type of screw they use, because you'll need matching machine screws (3/4" or 20mm long, quantity 8) to attach the counterweights to the hubs.

Parts list

- Motor (12VDC, dual-shaft, 4mm to 6mm shaft, power about 20W-50W, unloaded speed 3000 to 4500 RPM; see Chapter 61, Selecting a motor in Chapter 61, Shaker motors)
- Shaft hubs, with a central bore matching the motor shaft diameter, quantity 2
- Machine screws, sized to fit the threaded holes in the shaft hubs (typically M3 or #4-40), 3/4" (20mm) length, quantity 8
- Lock washers for the screws above, quantity 8 (optional)
- Machine screws, #10-32 x 1-1/2", quantity 8
- Machine screws, #10-32 x 3/4", quantity 4
- #10 lock washers, quantity 12 (optional)
- Hex nuts, #10-32 (preferably steel-with-nylon-insert lock nuts, also called ESN/elastic stop nuts), quantity 2
- Tee nuts, #10-32, quantity 10
- Fender washers, #10 x 1", stainless or zinc-plated steel, quantity approx. 20 (see notes below)
- Pipe hangers, quantity 2, or plumber's pipe hanger tape, sized to fit motor (see notes)
- Plywood, 1/2" thickness, about 1 sq ft
- Plywood, 3/4" thickness, about 1 sq ft

The fender washers are the main source of weight. The goal is to get the weight up to about 50g on each side. A typical 1" steel fender washer weighs about 5g, so about ten on each side should do the trick. Other than their weight, there's nothing special about the washers - I specified them because they're easy to attach with nuts and bolts. You can substitute another size of washer, or anything else of similar weight that you can figure out how to attach.

The #10-32 parts can be replaced by nearby sizes. Use whatever's most convenient. I used #10 for almost everything just to keep the number of distinct parts to a minimum.

The lock washers are optional, but I think it's a good idea to include them. They help

keep the screws from loosening over time due to vibration, and this thing's whole purpose is to generate a lot of vibration.

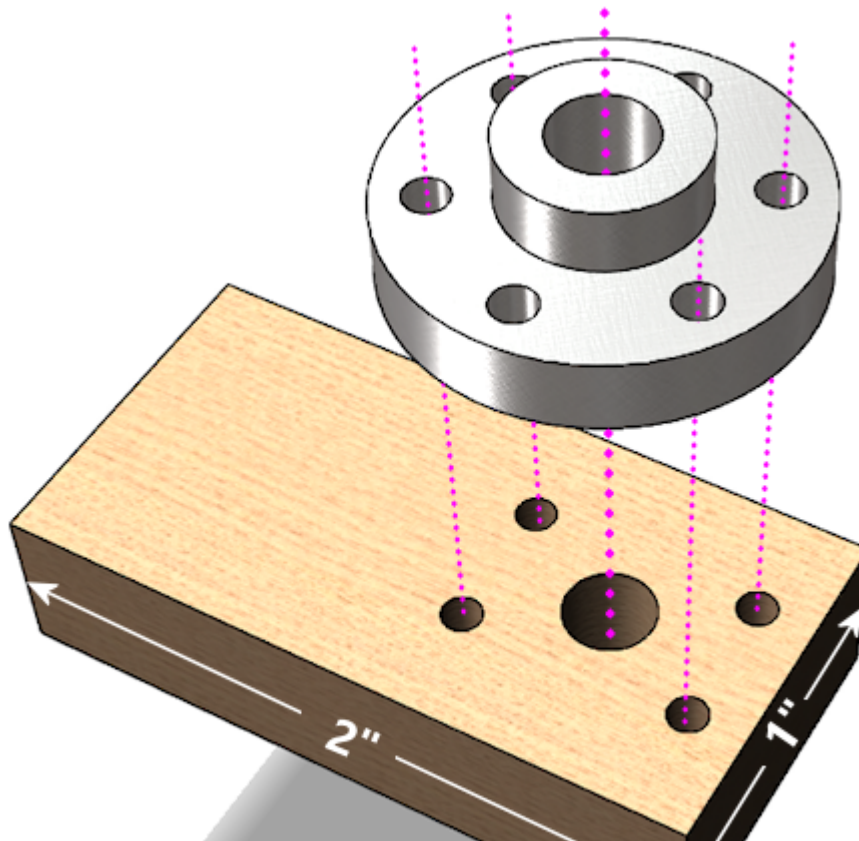
Pipe hangers are semicircular metal straps made to attach pipes to walls and ceilings. They come in various sizes from about 1/2" to 2", so you might be able to find one that's close to the size of your motor. You can find them at home improvement and hardware stores. Take your motor with you to check the fit - the nominal sizes are misleading because they're based on the *inside* diameter of the steel pipe they fit, so the actual size is always about 1/2" larger than the nominal size. If you can't find the right size, you can substitute plumber's pipe hanger tape, which is essentially a narrow strip of perforated sheet metal that you can cut to any length with sheet metal shears, and then bend around the motor to conform to the motor body's exact size. Be careful working with the tape - it's all sharp edges and jagged corners. Wear heavy leather work gloves.

Instead of the pipe hangers, you might be able to substitute U-bolts. U-bolts are easier to work with in some ways, but they're only available in certain sizes. My motor has a standard type "775" case, which is about 44mm in diameter; it fits perfectly into 1-3/4" U-bolts, specifically Bolt Depot #12489. You can also look for "exhaust clamps" or "muffler clamps" at an auto parts store - those are basically the same as U-bolts and might give you some additional size options, if you can't find a hardware store U-bolt in the right size.

Construction step-by-step

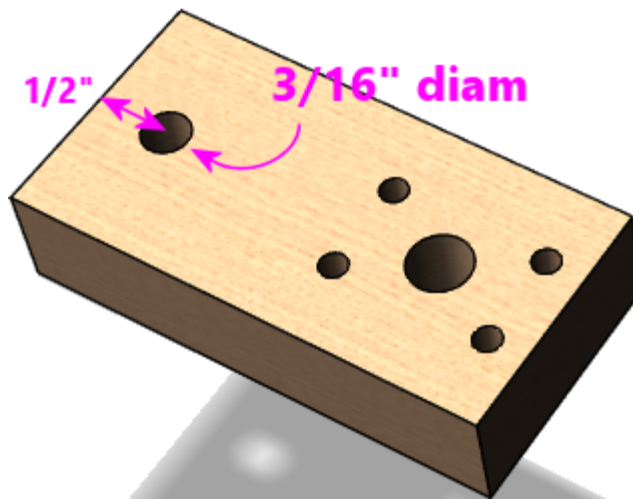
Step 1: Cut two pieces of the 1/2" plywood, about 2" x 1". These will be the "lever arms" for the counterweights.

Step 2: In each lever arm, near one end, drill a pattern of holes to match the holes in the hub for the central shaft opening plus two to four of the screw holes. Use the shaft hub as a template, and drill slightly larger than the holes in the hub, so that screws of the matching size will slide through freely. You really only need two screws for a strong attachment, even if your hub has four or six screw holes.

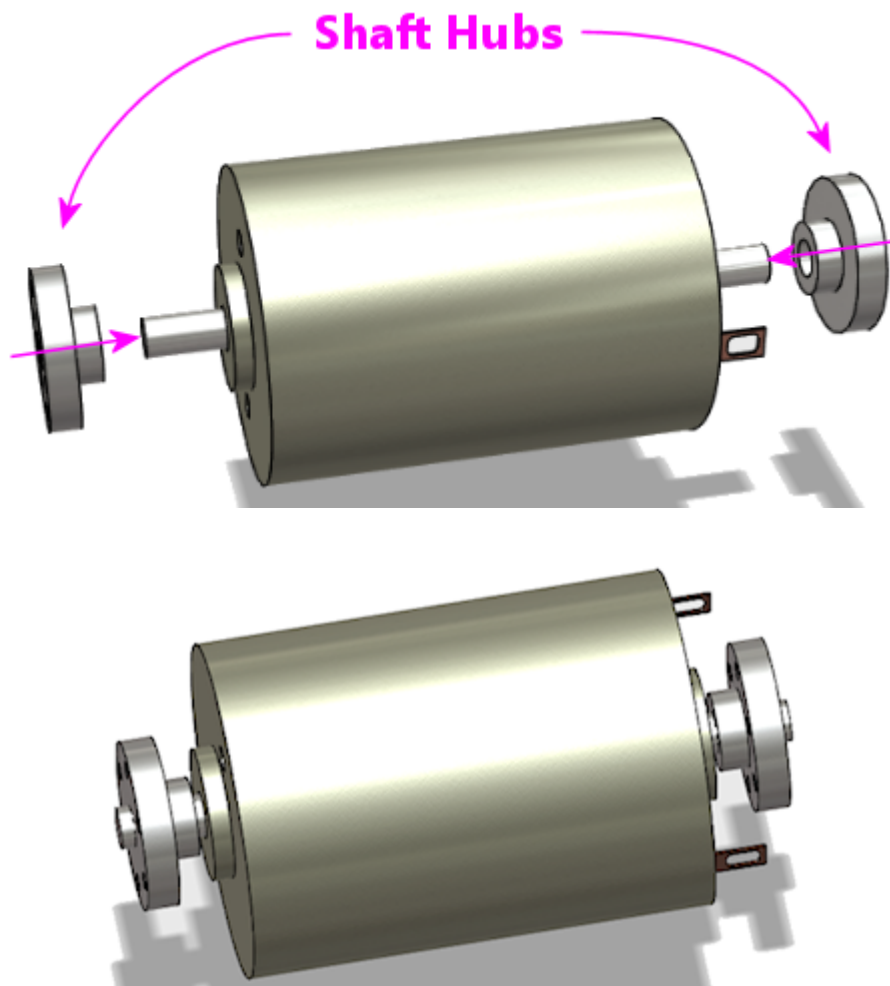




Step 3: Drill a 3/16" hole in the other end of each lever arm, with the hole center about 1/2" from the end.

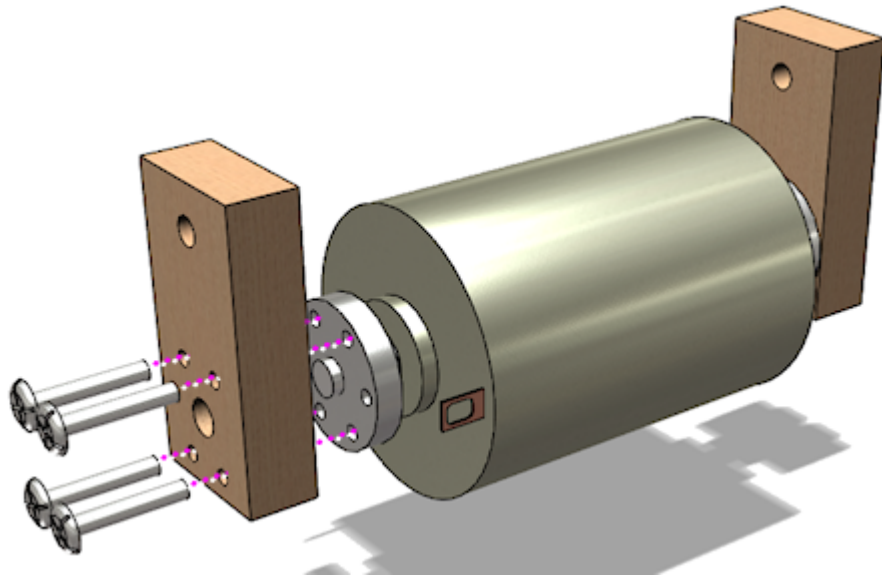


Step 4: Attach the shaft hubs to the motor, with the large disk sides facing out. Place them close to the motor body, but leave a little gap, so that they won't rub against the motor when spinning. Tighten the set screws.



Step 5: Attach the lever lever arms to the shaft hubs, using machine screws that fit

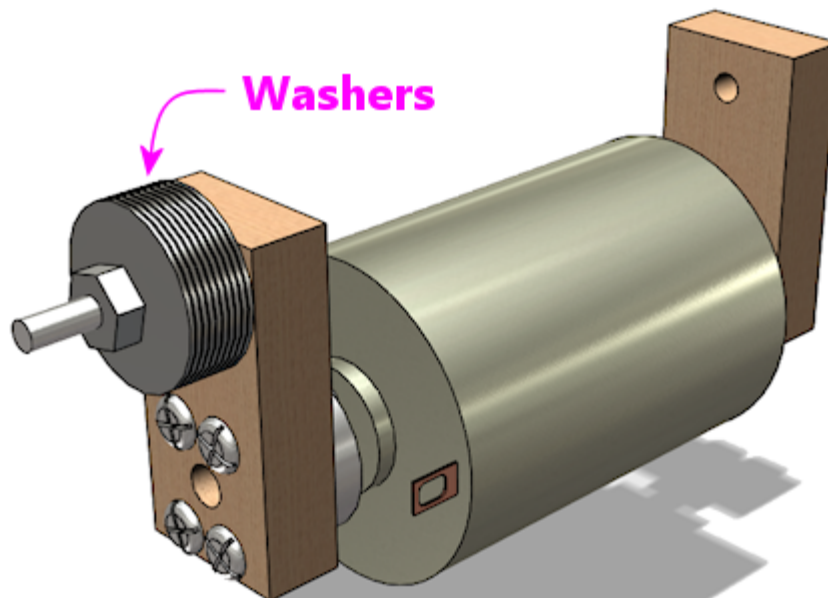
the shaft hub's threaded sockets.



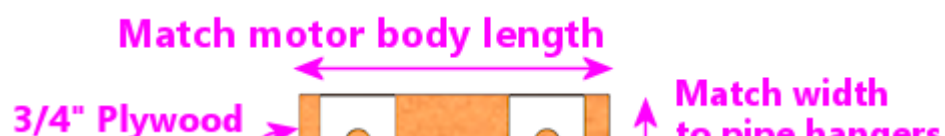
Note: I specified 3/4" length screws in the parts list, but those might be slightly too long or too short for some shaft hubs. If the fit is off (for example, if the screws stick too far out the other side of the hubs when tightened), you might need to substitute a different length.

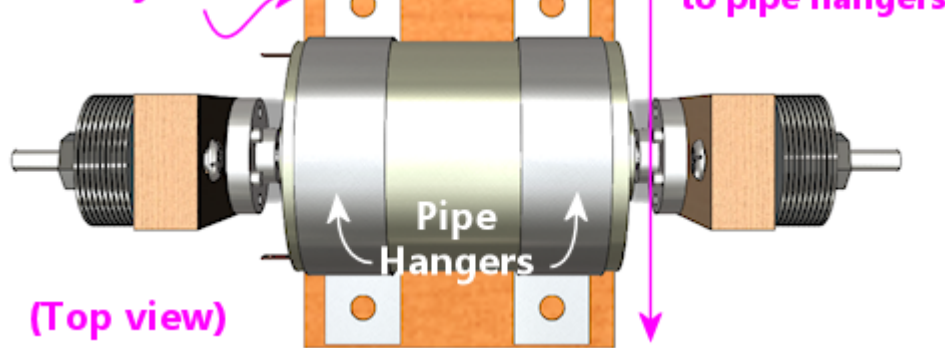
Step 6: If you have a kitchen scale or postal scale, weigh out about 50g worth of the fender washers for each side - this should be about 10 washers per side. Or you can just start with about 10 on each side and add or subtract some later if the shaking is too weak or too strong.

Step 7: Attach the washers to the lever arms using the #10 x 1-1/2" machine screws, lock washers, and #10 lock nuts.



Step 8: Cut a piece of the 3/4" plywood the same length as the central motor body, and slightly wider than the pipe hangers. This will be the upper base.



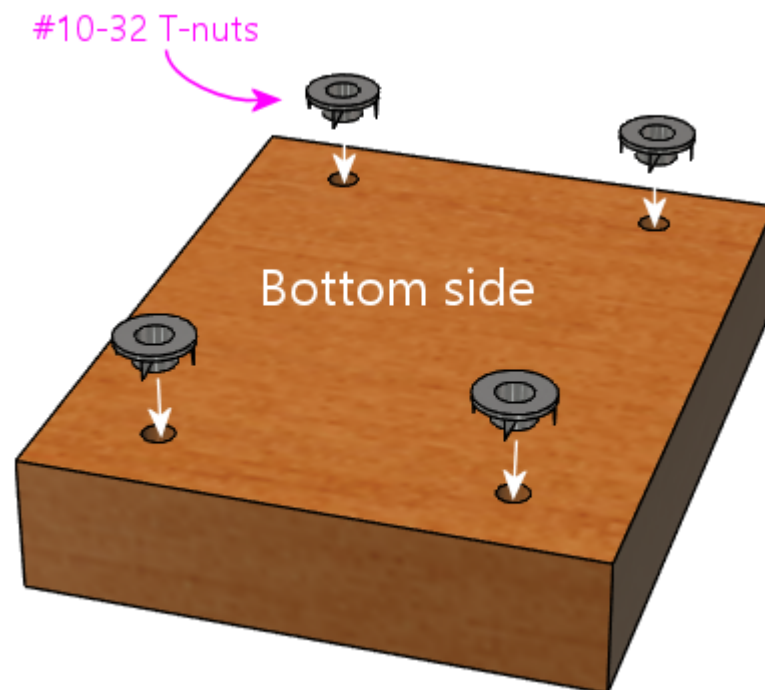


Step 9: Figure where to drill for the pipe hangers by placing the motor on the base, and fitting the pipe hangers over it, pressing them down tight over the motor. Mark the locations. Drill holes big enough for the #10 tee nuts.

It's critical to position the pipe hangers to make a very tight fit. The motor has to be completely immovable when the hangers are screwed down, so that it won't get dislodged by the shaking action. Stretch out the pipe hangers as necessary for a tight fit.

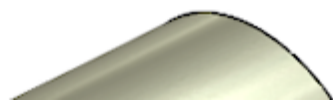
Note: Be careful not to cover any air vent openings in the motor when positioning the pipe hangers, and also keep them clear of the electrical terminals.

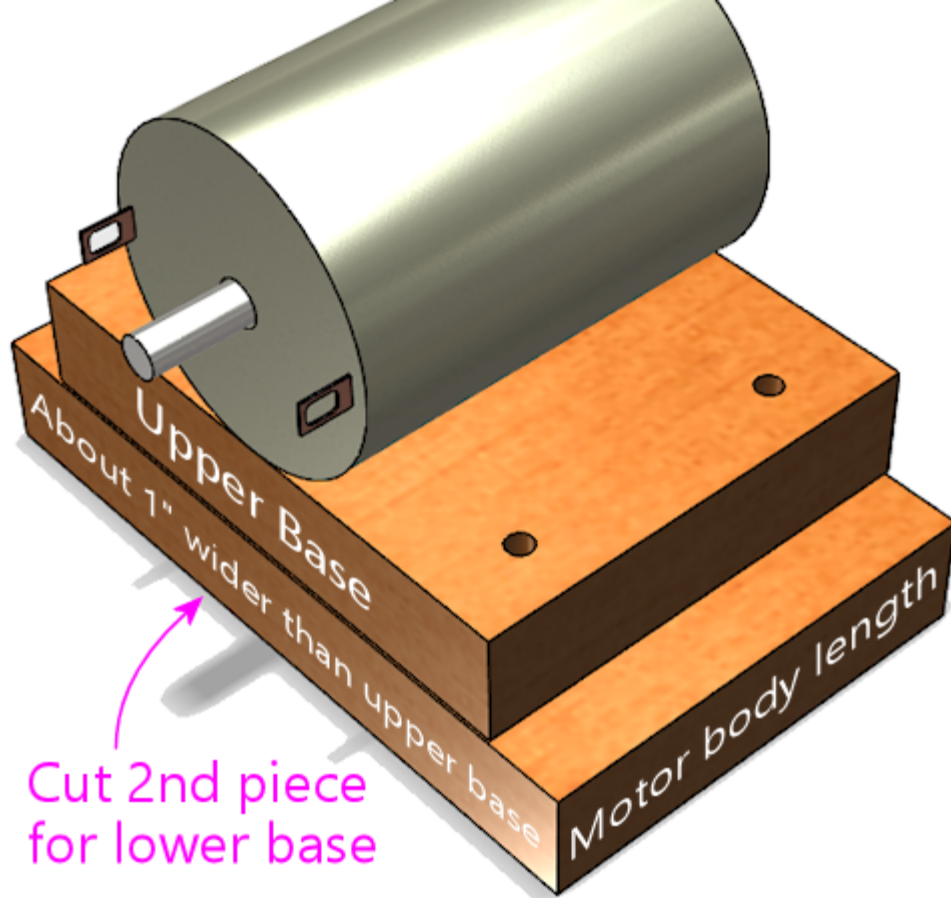
Step 10: Insert the #10 tee nuts into the holes from the bottom side of the base, and pound them in until flush.



The reason we're using tee nuts, by the way, is that this arrangement lets you attach and remove the motor straps even when the base plate is mounted to the cabinet floor. The tee nuts are permanently installed in the base plate, so the base plate effectively has threaded sockets for the motor fasteners.

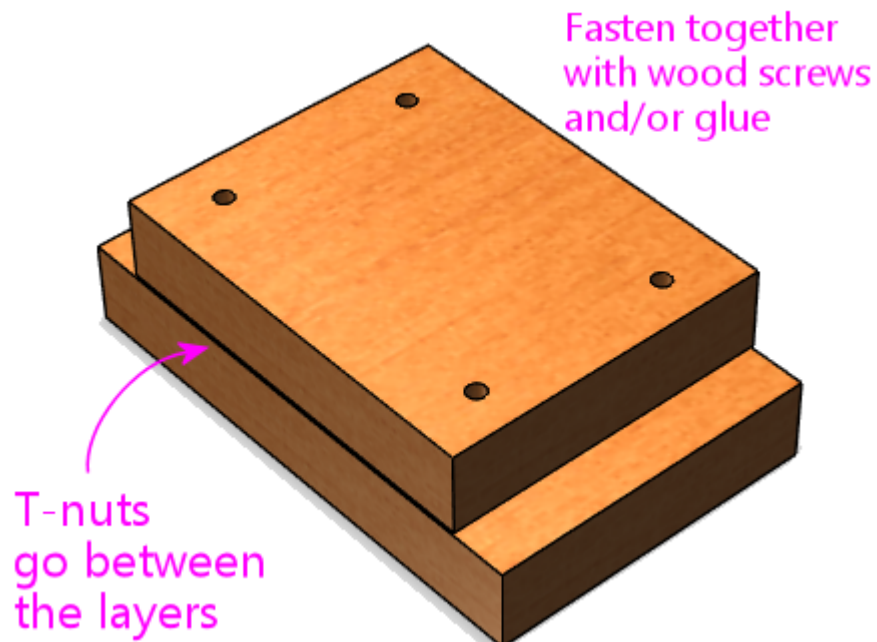
Step 11: Cut a second piece of 3/4" plywood, the same length as the first one, but about 1" wider. This will serve as the lower base.





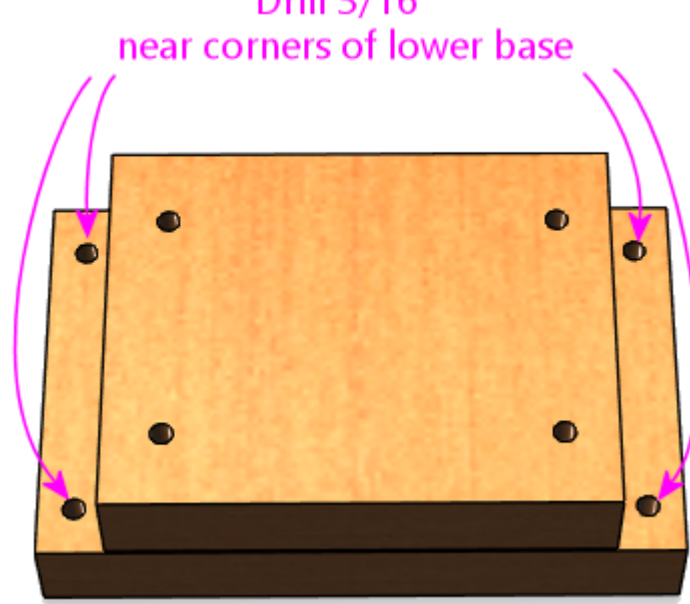
Step 12: Fasten the upper and lower bases together, with the upper base centered as shown. You can use a few wood screws and/or glue, as it's okay for this to be permanently attached. Make sure this is sturdy - it obviously has to stand up to the force of the shaking.

Important! Orient the top piece so that the tee nuts installed earlier are **between** the two layers.



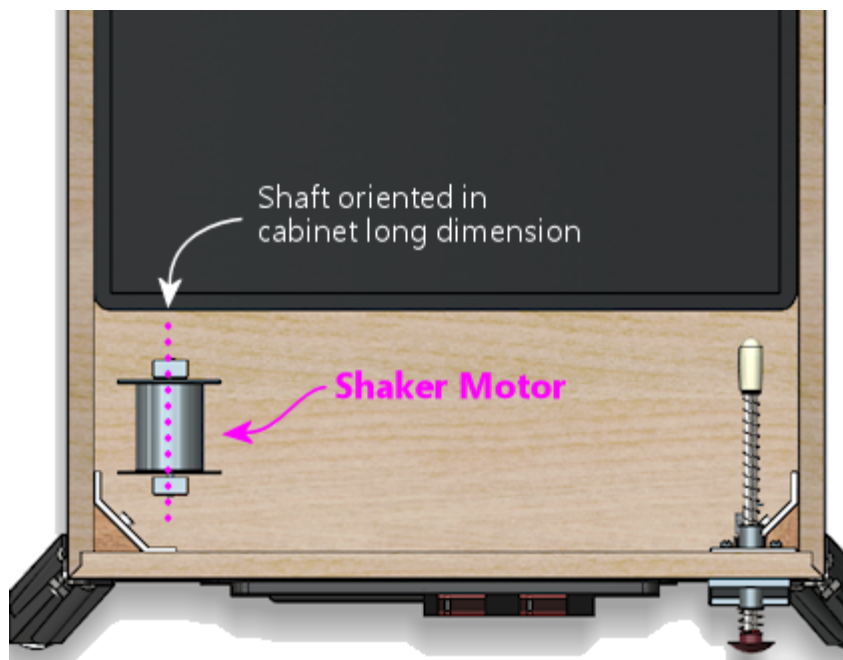
Step 13: Drill $\frac{3}{16}$ " holes at the corners of the lower base. These will be used to attach the base to the cabinet floor.

Drill $\frac{3}{16}$ "



Step 14: Figure where you want to install the assembly in the cabinet.

Shakers are usually installed somewhere towards the front of the cabinet, oriented with the motor shaft pointing along the **long** dimension of the cabinet, so that the shaking force is side-to-side. The exact location isn't critical, but closer to the front (and therefore closer to the player) seems to be better. It can be mounted close to one side or in the middle - I don't think it makes much difference.



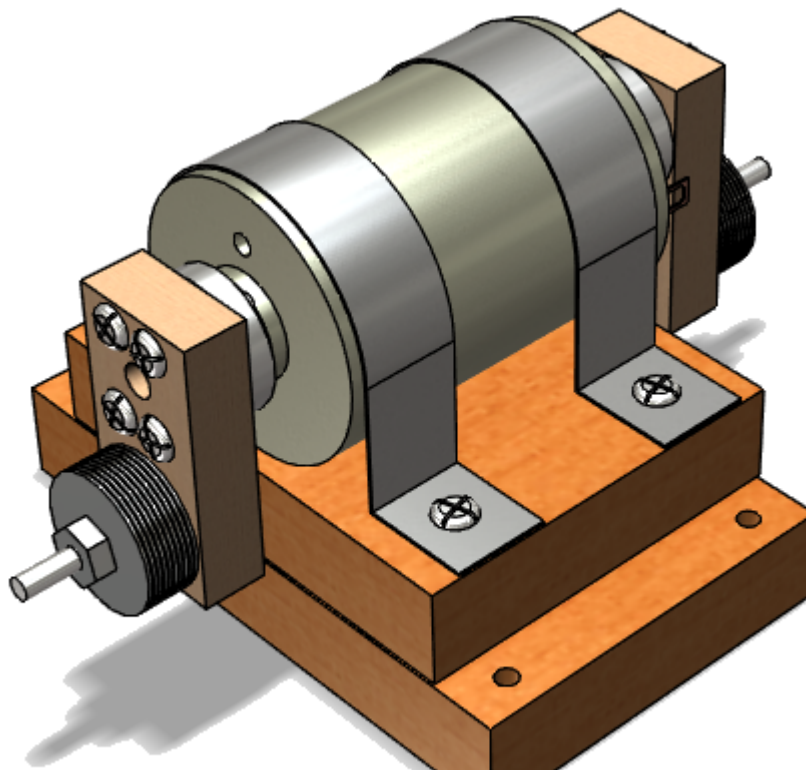
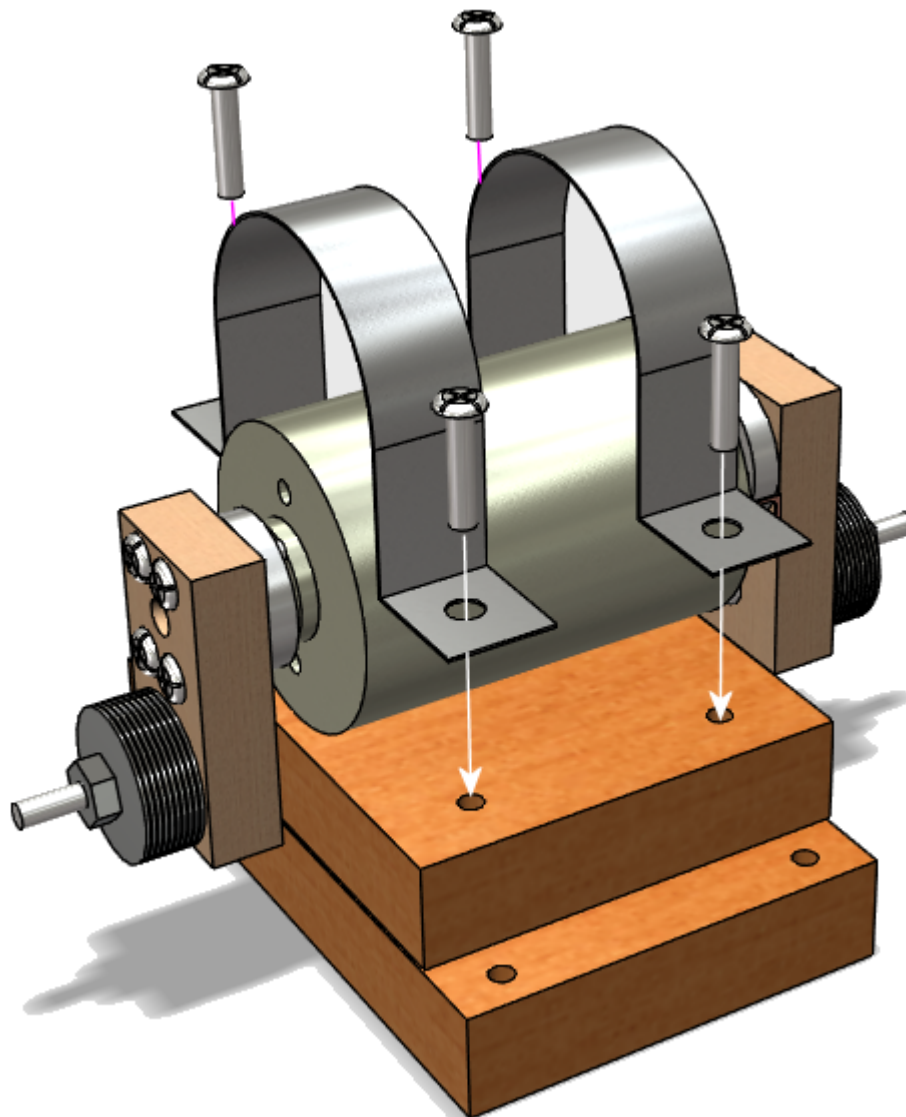
Step 15: Once you've determined the location, use the base as a template to mark the locations for the outer corner mounting holes on the cabinet floor. These are the holes in the **lower** base. Drill the marked holes in the cabinet floor - the drill size should be the same as the outer barrel diameter of your #10 tee nuts.


Step 16: Install tee nuts in the cabinet floor at the locations you just drilled. Insert the tee nuts from the **bottom** (outside) of the cabinet floor.

Step 17: Attach the motor to the base with the pipe hangers and the #10 x 3/4" screws and lock washers. The screws should mate with the tee nuts installed earlier.

Remember that the fit has to be extremely tight, so that the motor can't move at all.

If necessary, you can add something under the motor to fill any slack. I put some grippy rubber kitchen drawer liner under the motor in mine - that not only fills the space but also adds friction to keep the motor from rotating.





Step 18: Mount the base to the cabinet floor with #10 x 1-1/2" screws and lock washers. The screws mate with the tee nuts you installed in the cabinet floor earlier.

Secure the wiring

Make sure that the wires to the motor are routed so that they won't come into contact with the spinning weights. Secure them with cable ties as needed to make sure they stay that way.

Install a cover

A shaker motor should be fully enclosed with a sturdy cover, to keep fingers and loose parts away, and to contain ejected weights in case they come loose.

I'll leave the design of the cover up to you, but something like a plastic food storage container would work, or you could build a simple plywood box. Many people prefer a clear cover so that they can visually check the motor without taking the cover off.

End Notes

Acknowledgments

Many thanks are due to the members of the virtual pin cab forums at vpforums. My own build would never have gotten off the ground if the forums hadn't inspired me with lots of great examples of what you can accomplish, and I wouldn't have made it very far without the forums' collective expertise to draw on. This is one of the most helpful and on-topic online forums I've ever encountered, with a dedicated group of regulars who go way out of their way to help the newbies and solve the tough technical problems that inevitably come up in a project this complex. A lot of the information in the Cabinet Building section of this book came more or less directly from forum discussions; my main contribution here was to organize it into a more accessible format. As much as the forums are a gold mine of useful information, their free-form discussion format makes it extremely difficult to find specific topics that were discussed in the past. I hope that this guide is at least a little easier to navigate than old forum threads.

Another big thanks to the open-source developers who created and continue to improve the collection of software that makes virtual cabs possible: Visual Pinball, VPinMAME, DOF, and B2S. I especially want to thank them for keeping it all open-source. That helps ensure a strong future for these projects, by empowering their user communities to improve them and build upon them.

Colophon

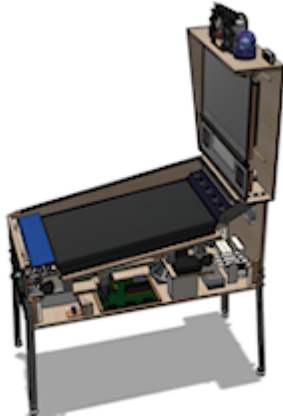
This book as you see it in your browser is generated from "source code" via a set of php scripts purpose-built for this application. The scripts take my source material, written in a simple ad hoc markup format, and generate the styled HTML.

The original motivation for the custom scripts was more than just for the sake of formatting. The real reason for custom scripts was that I planned to let you generate a highly customized subset of the material, based the unique combination of features you planned for your cabinet. The idea was that there are so many ways to build a virtual cab that it can get overwhelming to wade through the instructions for *everything*, so I thought I could organize it better by letting you set some filters to select only the features that were interesting to you. As I wrote the material, though, I could see that there's much, much more material that's common to all pin cabs than there is unique feature-based material, and I also realized that there are so many necessary cross-references that I couldn't safely filter out very much without a risk of links that go nowhere. Finally, when I stepped back and thought it about some more, I saw that this idea about filtering was a misguided attempt to reinvent a wheel. Book writers have been organizing complex material for centuries, and tools like the Table of Contents, Index, and cross-referencing are already known to work pretty well when applied competently.

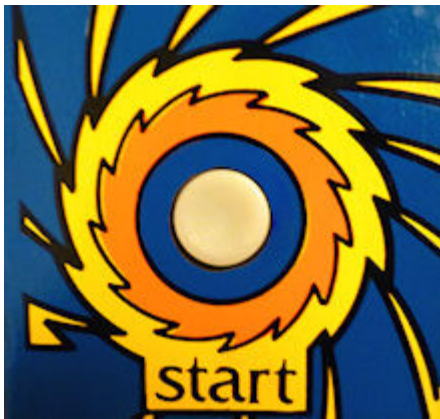
Most people (myself included) prefer to view technical material like this online, where it can be easily searched and where cross-reference links can be immediately followed, so my scripts are designed primarily with Web browser presentation in mind. However, I still have an old-fashioned affection for printed books, so I also tried to arrange things to produce pleasing hard-copy results. If you click the little Printer icon at the bottom, it will append all of the chapters together into a single (rather large) document that you can send to your printer to make a hard-copy edition.

A few people have asked about the CAD model that I used to generate illustrations throughout this book. Unfortunately, there's no practical way for me to share it, because it's based on an old proprietary software package that's no longer available, and it doesn't convert well to more modern formats (I've tried). Besides, most of the people who have asked were hoping to use some of the components for 3D-print manufacturing or at least for taking detailed measurements for planning purposes, but I'm afraid the model isn't accurate enough for those purposes. I only made it close enough to look about right in the illustrations.

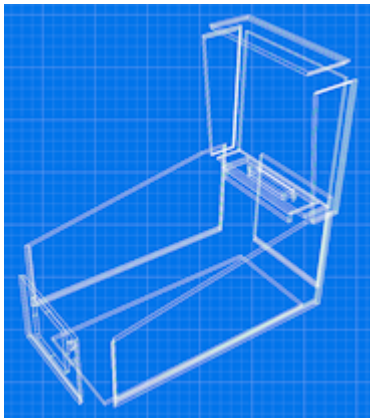
Notes on the section photos



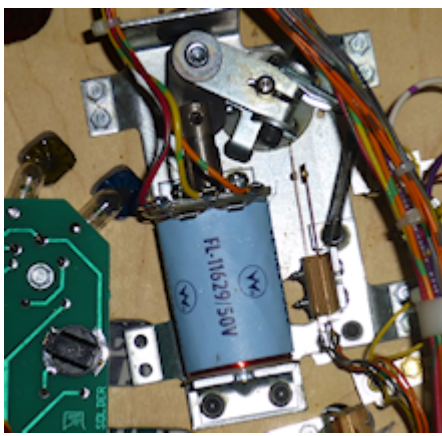
Cutaway view of a CAD model of a virtual pinball machine, used as the basis for many illustrations throughout this guide



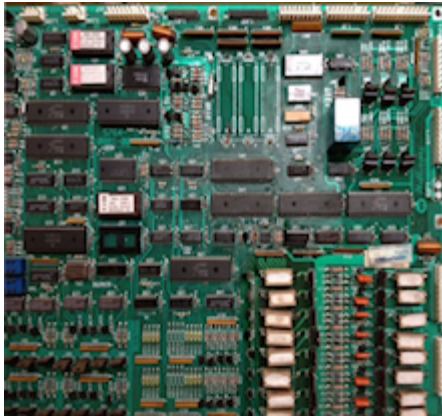
Start button from a *Whirlwind* pinball machine (Williams, 1990)



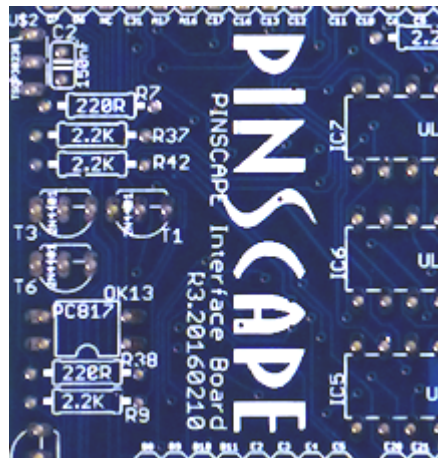
3D rendering of plans for a standard-body pin cab replicating the Williams/Bally "WPC" cabinet design of the 1990s



A flipper assembly (the parts under the playfield that actuate the flipper), from a Williams machine circa 1995



A Williams System 11 CPU board, which contains the main control electronics used in Williams pinball machines manufactured circa 1986-1990; in many ways, it amounts to a stripped-down 8-bit personal computer from the Apple II era



Closeup of an unpopulated main interface board for the Pinscape expansion boards



The Extra Ball buy-in button from *Theatre of Magic* (Bally, 1995)